

IBM i

*Remote System Explorer API  
Administration and Programming Guide*



**Note**

Before using this information and the product it supports, read the information in [“Notices” on page 111](#).

**Second Edition (May 2024)**

This edition applies to version IBM i 7.3 (product number 5770-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2023, 2024.**

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

# Preface

---

The *RSE API Administration and Programming Guide* describes the IBM® Remote System Explorer API, a collection of REST APIs that allow a client to work with various components on an IBM i host system, including QSYS objects, IFS files, jobs, and CL Commands.

## Who should read this book?

The *RSE API Administration and Programming Guide* is primarily for developers of web applications that need to access IBM i servers that use REST requests. Some of the information is useful to system administrators who manage systems.

## What you need to know to understand this book

Web application developers need to know JSON and how to invoke RESTful APIs. System administrators need to be authorized to use the IBM i Web Administration GUI to enable RSE API and troubleshoot problems.

## Conventions used in this book

*Italics* is used for new terms where they are defined.

Constant width is used for:

- Program language code listings
- JSON listings
- Command lines and options

*Constant width italic* is used for replaceable items in code or commands.

## Examples in this book

Examples used in *RSE API Administration and Programming Guide* are kept simple to illustrate specific concepts. Some examples are fragments that require other code to work.

## What has changed in this document

As new features and enhancements are made, the information in this document will get updated. To use any new features or enhancements you should load the latest HTTP Group PTF for your IBM i release. To see what HTTP Group PTF a feature or enhancement is in, go to the IBM Integrated Web Services for i Technology Updates wiki, at URL:

<http://www.ibm.com/developerworks/ibmi/techupdates/iws>

### Notes:

1. Sometimes new features or enhancements are not yet part of a group PTF, in which case the wiki will list the PTF number(s) containing the feature or enhancement.
2. To help you see where technical changes have been made since the previous edition, the character | is used to mark new and changed information.

The following lists the changes that have been made to the book since the previous edition:

#### • **May 17 2024**

- The Remote System Explorer (RSE) APIs have been enhanced to include new REST APIs to manage digital certificates and get information about TLS levels on the IBM i server. A new category of

services, called **Security Services**, has been created and contain the various security APIs. The new security APIs can be further sub-categorized as follows:

- **Digital certificate related APIs**

These APIs enables you to list certificates in certificate stores, get detailed information about certificates, delete certificates, import and export certificates. You can also change certificate store passwords.

- **Application definition APIs**

These APIs enables you to list application definitions, associate/disassociate certificates to/from application definitions. In addition, you can add and remove certificate authority (CA) digital certificates from the application definition CA trust list.

- **TLS APIs**

These APIs provide information about transport layer security (TLS) attributes and statistics for the system.

For more information, see [“API methods: Security Services” on page 47](#).

---

# Contents

<b>Preface.....</b>	<b>iii</b>
<b>RSE API overview.....</b>	<b>1</b>
Technical concepts.....	3
Overview.....	3
API categories.....	4
Getting started.....	5
Installation details.....	7
Starting and stopping the RSE API server.....	9
Starting and stopping the server by using CL commands.....	9
Starting and stopping the server by using Web Administration for i interface.....	9
Security.....	11
Transport level security.....	11
Authentication.....	12
Authorization.....	13
Administrator role.....	14
Security related properties.....	14
HTTP access log.....	15
Performance.....	17
Tuning RSE API.....	17
Tuning the network.....	17
Load balancing.....	18
Serviceability.....	19
Updates and fixes.....	19
Troubleshooting.....	19
Logging.....	19
JVM dumps.....	24
<b>RSE API reference.....</b>	<b>25</b>
API methods: Administration Services.....	27
GET /api/v1/admin/settings .....	27
POST /api/v1/admin/settings .....	28
GET /api/v1/admin/sessions .....	29
DELETE /api/v1/admin/sessions .....	30
GET /api/v1/admin/memory .....	30
GET /api/v1/admin/environment .....	31
API methods: CL Command Services.....	33
PUT /api/v1/cl .....	33
GET /api/v1/cl/{commandname} .....	34
API methods: IFS Services.....	37
GET /api/v1/ifs/list .....	37

GET /api/v1/ifs/{path} .....	39
PUT /api/v1/ifs/{path} .....	40
GET /api/v1/ifs/{path}/info .....	41
API methods: QSYS Services.....	43
GET /api/v1/qsys/search/{objectName} .....	43
API methods: Security Services.....	47
POST /api/v1/security/dcm/appdef/associate .....	47
POST /api/v1/security/dcm/appdef/disassociate .....	48
GET /api/v1/security/dcm/appdef/list .....	49
POST /api/v1/security/dcm/appdef/trust .....	50
POST /api/v1/security/dcm/appdef/untrust .....	51
POST /api/v1/security/dcm/cert/delete .....	52
POST /api/v1/security/dcm/cert/export .....	53
POST /api/v1/security/dcm/cert/import .....	54
POST /api/v1/security/dcm/cert/info .....	55
POST /api/v1/security/dcm/cert/list .....	57
POST /api/v1/security/dcm/certstore/changepassword .....	58
GET /api/v1/security/tls .....	59
GET /api/v1/security/tls/stats .....	61
API methods: SQL Services.....	65
PUT /api/v1/sql .....	65
API methods: Server Information Services.....	67
GET /api/v1/info/serverdetails .....	67
API methods: Session Services.....	69
GET /api/v1/session .....	69
PUT /api/v1/session .....	71
POST /api/v1/session .....	73
DELETE /api/v1/session .....	73
Components.....	75
Schemas.....	75
Security schemes.....	88
Configuration files.....	89
rseapi.properties.....	89
<b>RSE API Guides.....</b>	<b>91</b>
Configure TLS for the admin5 server.....	93
Testing RSE API by using OpenAPI UI.....	99
<b>Notices.....</b>	<b>111</b>
Trademarks.....	112
<b>Glossary.....</b>	<b>115</b>
<b>Index.....</b>	<b>119</b>

---

# RSE API overview

In support of web services and service-oriented architecture (SOA), the IBM i operating system integrates a software technology, IBM Remote System Explorer API (RSE API) that is a collection of APIs that allow a client to work with various components on the IBM i. This integration opens the IBM i system to various web service client implementations, including RPG, COBOL, C, C++, Java™, .NET, PHP, Enterprise Service Bus, mobile and web applications.

This part of the book introduces the RSE API concepts and architecture, including installation details.





# Technical concepts

The RSE API provides REST APIs that enables you to interact with IBM i objects such as database files, IFS files, and CL commands. RSE API requires a user to have a user profile on the target IBM i server.

An overview of RSE API is provided in the following sections.

## Overview

The RSE API web application runs in the admin5 integrated application server. The admin5 is one of several servers that fall under the *HTTP Administration Server* umbrella. The HTTP Administration Server consists of an instance of an IBM HTTP Server, named admin, and several instances of integrated application servers, named admin1, admin2, admin3, admin4, and admin5. The integrated application servers are used to run various IBM i web applications such as IBM Web Administration for i, which is a web application that is used to manage different types of servers, including admin5.

Figure 1 on page 3 shows the general flow between web clients and the RSE API application deployed in the application server.

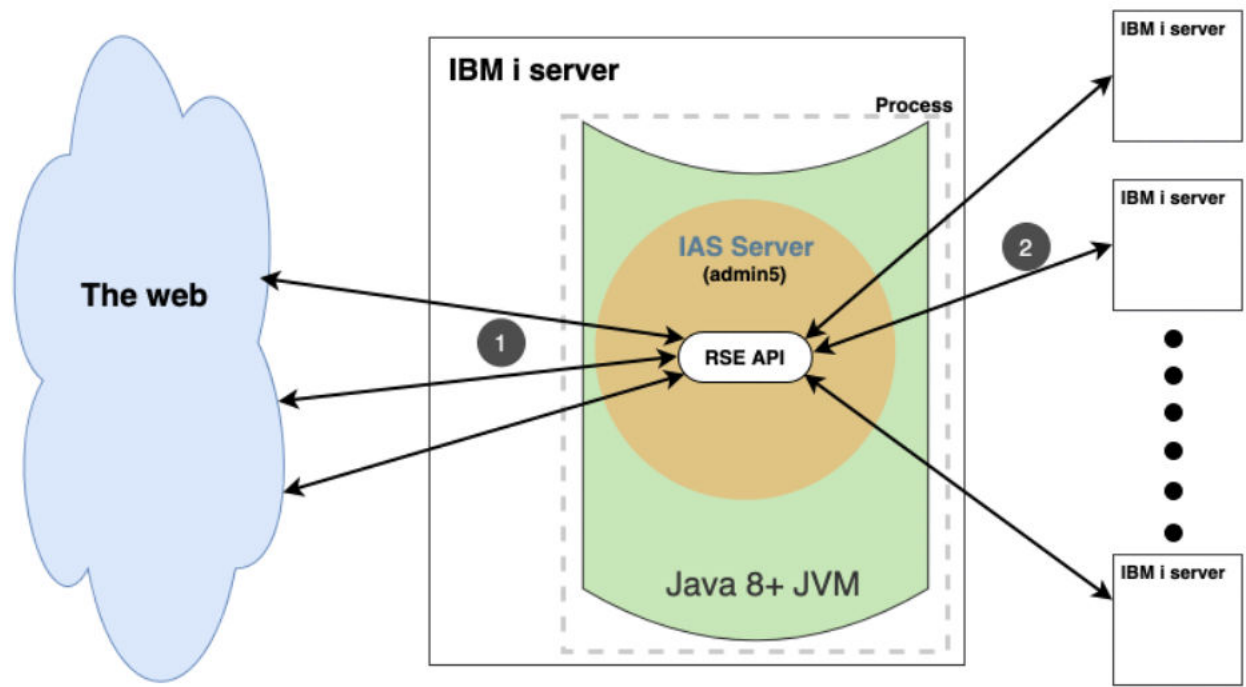


Figure 1. Flow between web clients and RSE API

The flow is as follows:

1. Clients make requests. Requests are routed to the RSE API application. All requests must be done over a secure connection (https). If transport level security (TLS) is not enabled, all requests are rejected.
2. The RSE API performs the requested operation and returns the response in JSON format. The target IBM i server can be a remote IBM i server or the IBM i server that the RSE API application is running on. Note that if indirectly accessing an IBM i server, TLS must be configured for the remote host server.

### RSE API and the Java Toolbox

The RSE API application can be thought of as an extension to the Java Toolbox classes, but instead of users having to write a Java application to access objects residing on an IBM i server that uses the Java Toolbox classes, users can invoke a REST API.

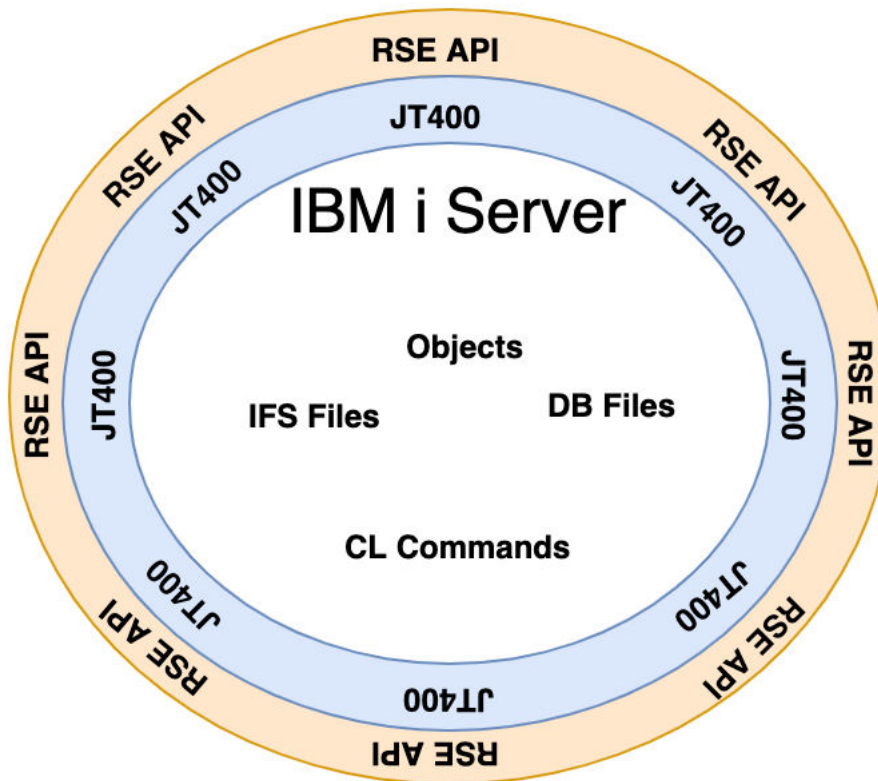


Figure 2. RSE API and Java Toolbox

Given that RSE API is heavily dependent on the Java Toolbox, any limitations in the Java Toolbox is applicable to RSE API.

## API categories

---

The RSE APIs can be categorized as follows:

- Administration Services

Get information about the state of the RSE API environment, such as number of sessions, memory usage, and setting global settings. Only users who connect to localhost and are designated as administrators or have \*ALLOBJ authority can successfully invoke the APIs.

- CL Command Service

The CL command service is used to run CL commands.

- IFS Services

The integrated file system (IFS) services are used to perform IFS operations such as reading from a file and writing to a file.

- QSYS Service

The QSYS service allows users to search for QSYS objects.

- Security Service

The security service APIs are used to manage digital certificates and to obtain system transport level security (TLS) information.

- SQL Service

The SQL service runs SQL statements.

- Server Information Service

The server information services retrieve information about the RSE API application.

- Session Services

Session services are used to authenticate and manage sessions.

Detailed information on the APIs can be found in [“RSE API reference”](#) on page 25.

## Getting started

---

Assuming that the IBM Web Administration Server active, you can view the online documentation by specifying the following URL: `http://host:2011/openapi/ui/`, where *host* is your IBM i host name or IP address. However, to use the APIs using the OpenAPI UI where you can view the request and response data without having to develop an application that uses the APIs, use the following URL: `https://host:2012/openapi/ui/`. If you need to enable TLS for the admin5 server, see [“Configure TLS for the admin5 server”](#) on page 93.

**Note:** The default ports for the admin5 server are 2011 (http) and 2012 (https). The ports can be different on your system depending on whether the system administrator configured ports other than the default.

The OpenAPI UI web page is generated that uses the OpenAPI specification (OAS), which defines a standard, language-agnostic interface to RESTful APIs, which allows one to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. See [“Testing RSE API by using OpenAPI UI”](#) on page 99 to learn how to use the OpenAPI UI.



# Installation details

This chapter describes the RSE API web application, including installation details and a description of the various components that make up the RSE API web application.

## Installing RSE API

The RSE API web application is included in option 3 of the base operating system. [Table 1 on page 7](#) lists prerequisite products that need to be installed:

<i>Table 1. RSE API prerequisite products</i>	
<b>IBM i OS version</b>	<b>Products</b>
7.3, 7.4, 7.5	<ul style="list-style-type: none"><li>• 5770SS1 option 3 - Extended Base Directory Support</li><li>• 5770SS1 option 12 - Host Servers</li><li>• 5770SS1 option 30 - Qshell</li><li>• 5770SS1 option 33 - PASE</li><li>• 5770SS1 option 34 - Digital Certificate Manager</li><li>• 5770DG1 - IBM HTTP Server for IBM i</li><li>• 5770JV1 - IBM Java SE 8 64</li></ul>
<b>Note:</b> After installing the various license product options, you should load the latest HTTP Group PTF since all fixes and enhancements are packaged as part of the HTTP Group PTF. It would also be wise to load the latest Java group PTF. See <a href="#">“Updates and fixes” on page 19</a> for further details.	

## Directory locations and properties

When the RSE API web application is installed, two directories are created as shown [Table 2 on page 7](#):

<i>Table 2. Install directories and contents</i>	
<b>Directory</b>	<b>Contents</b>
/QIBM/ProdData/OS/RSEAPI	Contains the RSE API package.
/QIBM/UserData/OS/RSEAPI	On initial install, the directory does not contain anything. When properties for the RSE API is configured, the properties will be stored in file /QIBM/UserData/OS/RSEAPI/rseapi.properties. For more information, see <a href="#">“rseapi.properties” on page 89</a> .



# Starting and stopping the RSE API server

The HTTP Administration Server server runs under the QHTTSPVR subsystem. If active, you will see various jobs as shown in Figure 3 on page 9.

```
Work with Active Jobs                                ZZ2P23
CPU %:      1.7      Elapsed time: 01:21:18      03/30/23 14:58:06 CDT
Active jobs: 176

Type options, press Enter.
 2=Change  3=Hold  4=End  5=Work with  6=Release  7=Display message
 8=Work with spooled files 13=Disconnect ...

Opt  Subsystem/Job  User      Type  CPU %  Function      Status
---  ---           ---      ---   ---    ---          ---
---  QHTTSPVR       QSYS      SBS   .0     DEQW          DEQW
---  ADMIN         QTMHHTTP  BCH   .0     PGM-QZHBMAIN SIGW
---  ADMIN         QTMHHTTP  BCI   .0     PGM-QZSRLOG  SIGW
---  ADMIN         QTMHHTTP  BCI   .0     PGM-QZSRHTTP SIGW
---  ADMIN1        QWEBADMIN BCI   .2     JVM-/QIBM/Prod THDW
---  ADMIN2        QLWISVR   BCI   .1     JVM-/QIBM/Prod THDW
---  ADMIN3        QWEBADMIN BCI   .1     JVM-/QIBM/Prod THDW
---  ADMIN4        QWEBADMIN BCI   .1     JVM-/QIBM/Prod THDW
---  ADMIN5        QLWISVR   BCI   .1     JVM-/QIBM/Prod THDW
```

Figure 3. HTTP Administration Server jobs

The jobs with the name of admin are the HTTP Server jobs, the other jobs (admin1-admin5) are the integrated application server jobs.

In the following sections, the CL commands that are used requires that user to have authority to the command.

## Starting and stopping the server by using CL commands

You can start the HTTP Administration Server by using the following CL command:

```
QSYS/STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

When the server starts, not all integrated application server jobs may be active. A system administrator is able to disable a server by using the Web Administration for i GUI interface. If that is the case for admin5, ask the system administrator to enable the server. After the server is enabled, restart the HTTP Administration Server or start the admin5 server by using the following CL command:

```
QSYS/STRTCPSVR SERVER(*IAS) INSTANCE(admin5)
```

End the HTTP Administration Server by using the following CL command:

```
QSYS/ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

To end the admin5 server, the following CL command can be used:

```
QSYS/ENDTCPSVR SERVER(*IAS) INSTANCE(admin5)
```

## Starting and stopping the server by using Web Administration for i interface

Various types of servers are managed through the Web Administration for i interface. The Web Administration for i interface is a browser-based application that is loaded in the HTTP Administration Server and is typically accessed by using the following URL: `http://<host>:2001/HTTPAdmin`, where *host* is the host name or IP address of the IBM i server.

**Note:** If you are unable to connect to the server, check to see whether the HTTP Administration Server is active.

The Web Administration for i interface combines forms, tools, and wizards to create a simplified environment to set up and manage many different servers on your system. The wizards guide you through a series of advanced steps to accomplish a task.

It is assumed that you are familiar with the Web Administration for i interface. If not, read about the interface under the HTTP Server topic in the *IBM Documentation* at [http://www.ibm.com/support/knowledgecenter/ssw\\_ibm\\_i](http://www.ibm.com/support/knowledgecenter/ssw_ibm_i).

By default, only users with \*ALLOBJ and \*IOSYSCFG special authorities can manage and create web-related servers on the system by using the Web Administration for i interface. A user without the necessary IBM i special authorities to manage or create web-related servers requires an administrator to grant that user permission to a server or group of servers.

To start and stop the admin5 server, navigate to the admin5 server by clicking the **Application Servers** tab and choosing the admin5 server from the **Server** selection box as shown in [Figure 4](#) on page 10.

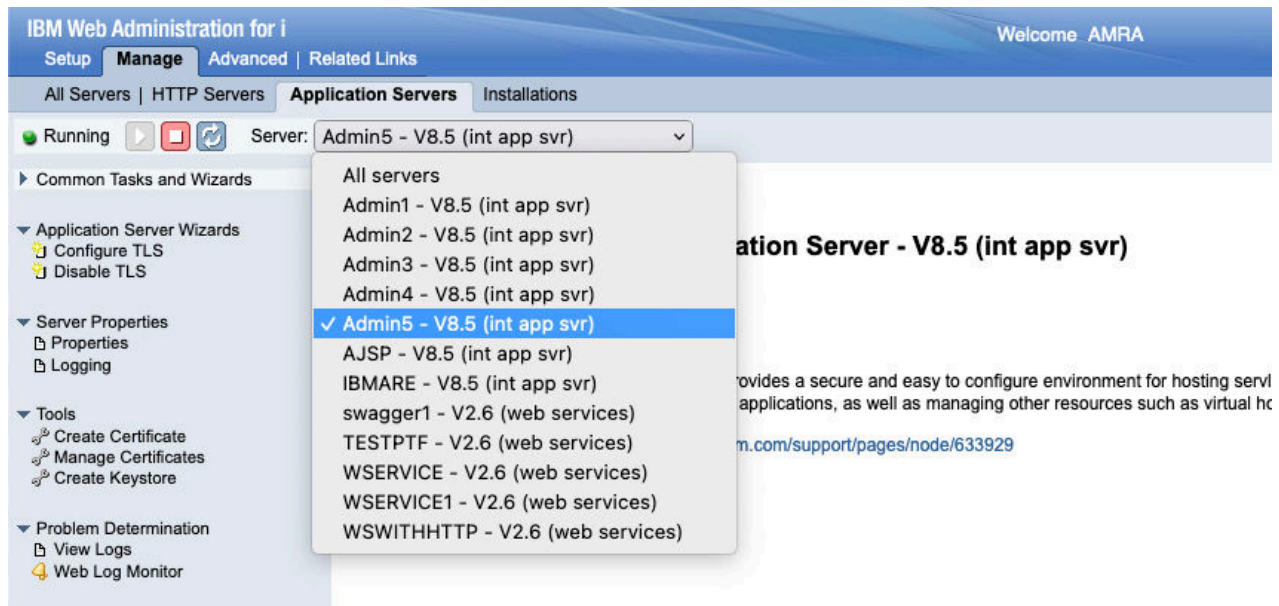


Figure 4. Selecting an application server

Once a server is selected, the admin5 is shown and you can start or stop the server by using the buttons on the page.



---

# Security

This chapter discusses various topics that are related to security as it pertains to the usage of RSE API.

---

## Transport level security

*Transport level security* (TLS) is a communication protocol that is used to secure browser to server, and client to server traffic across the internet and internal networks.

All RSE API requests must be submitted over a secure connection (`https`). An API request over an unsecure connection will fail. If indirectly accessing an IBM i server, the host servers on that server need to be configured for TLS to accept connections over secure connections.

TLS is a useful tool to secure communications, but it must be deployed correctly otherwise the data in transit is vulnerable to modification or eavesdropping and the TLS merely provides a false sense of security. Occasionally TLS is incorrectly referred to as 'SSL', which is an earlier protocol deprecated due to security vulnerabilities. TLS uses public key cryptography to exchange keys between client and server, symmetric encryption to encrypt data in transit and x.509 certificates to identify the server to the client, and in some configurations identify the client to the server.

The following steps are essential to deploy a secure TLS connection:

- Securely generate and store a private key.
- Obtain a certificate associated with the private key, signed by the appropriate trusted Certificate Authority (CA).
- Configure the server to use most secure TLS versions (for example, 1.2 or 1.3, or more secure version as new TLS versions become available) and approved cipher suites.
- Configure the client to check certificate validity and use approved TLS versions.
- Deploy certificate and private key to server.
- Replace the certificate before it expires (typically certificates expire after 1 year).

Each of these steps has its own security requirements that need to be considered, and are described in the following sections.

### Private key recommendations

The following are private key recommendations:

1. RSA key length should be at least 2048, and 4096 or stronger is recommended, ECC key length must be at least 224, 256 recommended.
2. Private keys should not be sent in unencrypted emails or instant messages.
3. Private keys should not be stored in github or other source control systems.
4. Private keys should not be stored unencrypted on engineers' workstations or devices.
5. Access to private keys should be restricted to the engineers and systems that require access, using role based access control, access control lists or an equivalent control.

### Certificate recommendations

The following are certificate recommendations:

1. All external facing certificates used should be issued through an approved Certificate Authority (CA).
2. Certificate private keys must be unique to that certificate. Private keys must not be re-used for any other purpose.
3. Whenever a certificate is renewed, a new private key should be used.

4. External facing certificates issued by an external CA should have a maximum lifetime of 371 days (that is, 1 year, plus 6 days to allow procurement/deployment).
5. Internal facing certificates issued by an internal CA for service to service connections should have a maximum lifetime of 2 years.
6. Internal facing certificates issued by an internal CA for browser to service connections should have a maximum lifetime of 1 year.
7. Internal Online CA signing certificates should have a maximum lifetime of 10 years. Internal Offline CA certificates and root certificates should have a maximum lifetime of 10 years
8. Certificates should use a SHA2-256 (SHA256) signature or stronger. SHA-1 signatures should not be permitted.

## TLS protocol recommendations

The following are TLS protocol recommendations:

1. Only TLS versions 1.2 or 1.3 should be permitted. All older versions of TLS and all version of SSL must not be used.
2. Only permitted cipher suites should be used.

## Cipher suites recommendations

The following are cipher suites recommendations:

1. Minimum RSA key size should be 2048, and 4096 is recommended.
2. Minimum AES key size should be 128.
3. Minimum ECDSA key size should be 224, and 256 is recommended.
4. Minimum ECDHE key size should be 224, and 256 is recommended.
5. SHA2 or SHA3 should be used for hash algorithm. SHA1 or MD5 must not be used.

## Authentication

---

RSE API handles authentication to the REST APIs. Access to any objects or information on a server is done by using the Java Toolbox. Thus, a user must have a user profile to the requested server.

The authentication schemes that are supported by RSE API are:

- HTTP bearer authentication
- HTTP basic authentication

All valid authentication requests are audited and logged.

### HTTP bearer authentication

An HTTP bearer authentication scheme is used to generate access tokens that are exchanged between the server and the client when calling the API operations. The token is exchanged between the client and the server in the `Authorization` HTTP header in the following format:

```
Authorization: Bearer token
```

For RSE API to be used, authenticate by using the POST HTTP method session API. The request body format contains the host, user ID, and password (for more information, see [“POST /api/v1/session” on page 73](#)). An example of the payload:

```
{
  "host": "localhost",
  "userid": "user",
  "password": "pwd"
}
```

If authentication is successful, a token is returned in the Authorization HTTP header. An example of an HTTP header:

```
Authorization: Bearer 4eaa14e6-ea9c-4dde-b6f7-f542b34d5309-60da75d3-3132
```

The client must then send the token in the Authorization HTTP header on all API requests.

## HTTP basic authentication

HTTP basic authentication is a simple challenge and response mechanism with which a server can request authentication information (a user ID and password) from a client. The client passes the authentication information to the server in an Authorization header. The authentication information is in base-64 encoding. The format of the Authorization HTTP header is as follows:

```
Authorization: Basic BASE64(userid:password)
```

When a client request using HTTP basic authentication is authenticated, the API is then invoked and the response is returned. A token is not returned.

**Note:** The session APIs do not support HTTP basic authentication.

## Authentication errors

All valid authentication requests are audited and logged to the messages .log file. If an authentication is successful, you will see an entry in the log file that is similar to the following example:

```
[4/14/23 13:21:06:002 CDT] 00000291 IBMIRSEAPI A login: User 'amra' authenticated.
```

If authentication fails, the response that is returned is similar to the following response:

```
{
  "title": "UNAUTHORIZED",
  "status": 401,
  "detail": "User ID or password is not set or not valid.",
  "method": "GET",
  "instance": "/rseapi/api/v1/ifs/list",
  "timestamp": "2023-04-14T18:26:39.961Z"
}
```

If you want to find out the reason for the failure, you will need to check the log file. Authentication failures are logged with detailed information on the authentication failure. For example,

```
[4/14/23 13:26:39:961 CDT] 00000291 IBMIRSEAPI A login: Login attempt by user 'amra' failed.
Password is incorrect.:AMRA
```

For more information at looking at server log files, see [“Logging” on page 19](#).

## Authorization

---

Every system user must have a user identity before they can sign on to and use a system. This user identity is a special object that is called a *user profile*.

A user profile is a string of characters that uniquely identifies a user to a system and controls what the user can do and access to functions and objects to which they have been granted authority. Only an administrator with appropriate system authority can create a user profile. (In this document, user ID and user profile is used interchangeably.)

Authorization defines the content that you can view and the actions that you can perform. Thus, the user profile that is used to authenticate to RSE API governs what a user can do when attempting to perform a request to an API.

When an API returns a list of objects, some attributes of the object can be set to null depending on data and object authorities the user has been given to the object. In the following response returned by the IFS list API, the object represented by path /QSYS.LIB/NOAUTHLIB.LIB has various attributes set to null:

```
{
  "objects": [
    {
      "path": "/QSYS.LIB/NETDATACOL.LIB",
      "description": null,
      "isDir": null,
      "subType": null
    },
    {
      "path": "/QSYS.LIB/NETDATADEV.LIB",
      "description": "",
      "isDir": true,
      "subType": "PROD"
    },
    {
      "path": "/QSYS.LIB/NOAUTHLIB.LIB",
      "description": null,
      "isDir": null,
      "subType": null
    }
  ]
}
```

In cases where an attempt is made to access the object for which the user has no authority, an error is returned. The following is an example of an error that is returned when trying to access the object for which the user has no authority:

```
{
  "title": "FORBIDDEN",
  "status": 403,
  "detail": "CPF9820 Not authorized to use library NOAUTHLIB.",
  "method": "GET",
  "instance": "/rseapi/api/v1/ifs/list",
  "timestamp": "2023-04-14T17:52:12.441Z"
}
```

See [“RSE API reference”](#) on page 25 for a description of the fields in the error response.

## Administrator role

---

RSE API has an administrator role. Any user profile with \*ALLOBJ special authority or has been designated as an administrator will be able to use the Administration Services APIs.

The Administration Services APIs enable a user to obtain information about the server hosting the RSE API web application, such as memory usage, and active sessions per user. You can also invalidate sessions. More importantly, the APIs enable an administrator to set global attributes, such as the maximum number of sessions, and the ability to restrict who can use RSE API.

For more information on what you can do with the Administration Services APIs, see [“API methods: Administration Services”](#) on page 27.

## Security related properties

---

RSE API has a set of properties to protect against denial of service (DoS) attacks or compromising system resources.

- com.ibm.rseapi.excludeusers
- com.ibm.rseapi.includeusers
- com.ibm.rseapi.maxfilesize
- com.ibm.rseapi.maxsessioninactivity
- com.ibm.rseapi.maxsessionlifetime

- com.ibm.rseapi.maxsessions
- com.ibm.rseapi.maxsessionsperuser
- com.ibm.rseapi.maxsessionusecount
- com.ibm.rseapi.maxsessionwaittime

See [“rseapi.properties”](#) on page 89 for further details.

## HTTP access log

---

Logs are granular, timestamped, complete, and immutable records of application events that can be used for troubleshooting and debugging purposes.

HTTP access logs record information about all requests that are handled by a web server. The access log is highly configurable, and you define the contents and format of the access log. The log can include information about the IP address of the client making the request, user ID of the person making the request (determined by the HTTP authentication), timestamp when the request was received, the request line, and so on.

HTTP access logs should be periodically analyzed to ensure that no unusual activity is taking place. For example, if you have 100,000 requests in 5 minutes from one specific IP address. HTTP access logs can also be used to detect API problems, performance or otherwise. Finally, and most importantly, logging allows one to backtrack and do forensic analysis when an intrusion is detected.

You should consider turning on HTTP access logging for the `admin5` server. By default, HTTP access logging is disabled. See [“HTTP access logging”](#) on page 22 for further details.



---

## Performance

When looking at improving performance, one needs to look at the entire *web environment*. In this context, a web environment is a grouping of related web server, application server, and operating system settings that form a web solution. This chapter attempts to highlight various approaches to improving the performance of RSE API. The information in this section share common approaches to solving common problems based on real environments. They do not provide a “one size fits all” solution. As technology evolves, new recommendations and information might be added to the information in this document.

For a definitive discussion on performance, you should read the topics in IBM Documentation under the performance topic on the [Performance](#) web page.

A key reference from which the information in this chapter is derived from is the *IBM WebSphere Application Server Performance Cookbook* at <https://publib.boulder.ibm.com/HTTPSERV/cookbook/>. Although the cookbook covers performance tuning for WebSphere® Application Server, there is a very strong focus on Java, Operating Systems, and theory that can be applied to other products and environments.

---

## Tuning RSE API

RSE API is a Java web application and thus runs in a JVM in the admin5 server job. The JVM is configured to have a maximum Java heap size of 4096 megabytes and generates `OutOfMemory` errors when there is insufficient space to allocate an object in the Java heap. In RSE API, this can be manifested by a combination of one or more of the following actions:

- High number of threads (that is, sessions) handling API requests, which can occur when there are thousands of simultaneous RSE API requests.
- Large amount of incoming or outgoing data, such as when writing or reading to an IFS file.
- Large amount of IBM i objects processed, such as performing a search that results in thousands of objects being processed.

To mitigate the occurrence of `OutOfMemory` errors and the performance of RSE API, careful consideration must be taken when setting the following RSE API properties:

- `com.ibm.rseapi.excludeusers`
- `com.ibm.rseapi.includeusers`
- `com.ibm.rseapi.maxfilesize`
- `com.ibm.rseapi.maxsessioninactivity`
- `com.ibm.rseapi.maxsessionlifetime`
- `com.ibm.rseapi.maxsessions`
- `com.ibm.rseapi.maxsessionsperuser`
- `com.ibm.rseapi.maxsessionusecount`
- `com.ibm.rseapi.maxsessionwaittime`

Further details on these properties can be found in [“rseapi.properties”](#) on page 89.

---

## Tuning the network

The network adapters that are used determine the maximum speed that can be reached. For example, 1 Gb, 10 Gb, and so on (at least in theory). However, the protocol being used, the networking parameters set, the quality of the connections, as well as other systems in the same subnet in the network determines the actual performance in terms of network throughput and speed.

- Consider increasing the TCP/IP buffer for send and receive operations by using the CHGTCPA CL command to a value greater than the default of 64 KB. This can significantly reduce the network traffic.
- Ensure that the parameters for current line speed is reflecting what the adapter is capable of. A single device with a lower line speed capability forces the whole subnet in the network to run at the lower speed and can severely degrade network performance.
- Ensure the current DUPLEX parameter is set to \*FULL so the connection can be used for send and receive at the same time.
- Consider increasing the maximum frame size from 1496 bytes to 8996 bytes as this can significantly reduce the traffic between the system and the next network router and speed up the connections especially when virtual Ethernet connections are used.

## Load balancing

---

If you have multiple partitions or servers, you can have a load balancer spray requests among the RSE API servers. However, the load balancer must be smart enough to assign subsequent requests to the same server since tokens are not shared between the servers.



# Serviceability

A system administrator must be able to diagnose and troubleshoot issues in the RSE API web application. This chapter describes various methods that can be used to resolve problems that can be encountered while using the RSE API web application.

## Updates and fixes

The latest RSE API PTFs for the various supported releases are documented on the RSE API *Technology updates* at <https://www.ibm.com/support/pages/node/6982713>.

Eventually, PTFs are included in the latest HTTP group PTF that corresponds to the IBM i operating system release. The various group PTFs for an IBM i release can be found at *IBM i Group PTFs with level* at <https://www.ibm.com/support/pages/ibm-i-group-ptfs-level>.

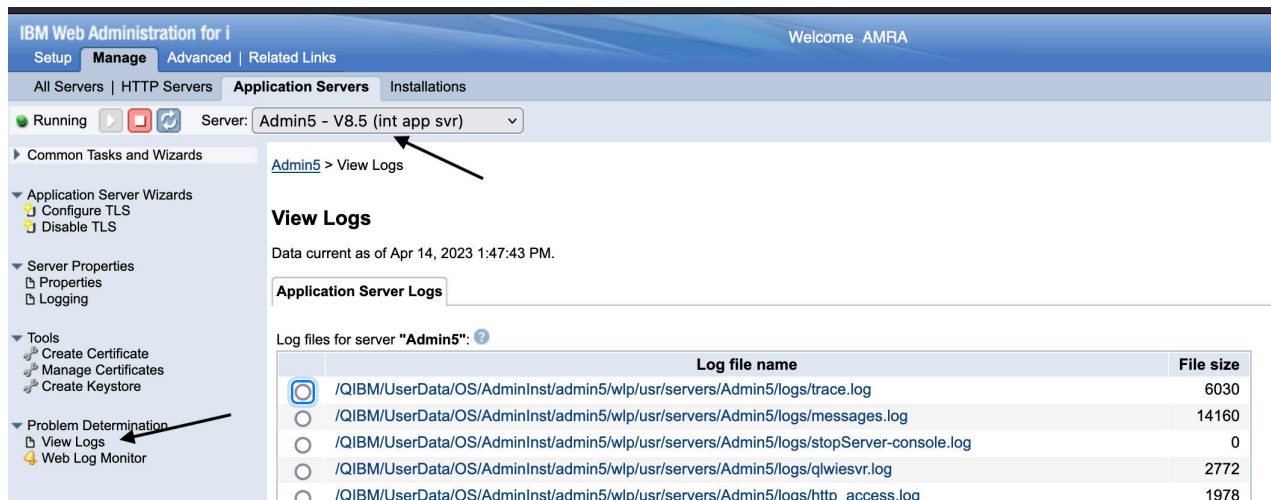
## Troubleshooting

There are several tools to help identify problems with the server or the RSE API web application.

### Logging

The integrated application server can produce a variety of traces that help to debug issues with the server and applications.

The integrated application server records limited information by default. This basic information is useful for debugging common configuration issues. You can view the output logs for the `admin5` server by opening the messages .log file by clicking the **View Logs** link in the navigation bar (see [Figure 5 on page 19](#)).



The screenshot shows the IBM Web Administration for i interface. The top navigation bar includes 'Setup', 'Manage', 'Advanced', and 'Related Links'. The main navigation bar shows 'All Servers | HTTP Servers | Application Servers | Installations'. The 'Application Servers' section is active, showing a server named 'Admin5 - V8.5 (int app svr)'. A dropdown menu is open, showing 'Admin5 > View Logs'. The 'View Logs' page displays the date 'Apr 14, 2023 1:47:43 PM' and a table of log files for the 'Admin5' server. The table has columns for 'Log file name' and 'File size'. The log files listed are: '/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/trace.log' (6030), '/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/messages.log' (14160), '/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/stopServer-console.log' (0), '/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/qlwiesvr.log' (2772), and '/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/http\_access.log' (1978). A sidebar on the left contains various navigation options, including 'Common Tasks and Wizards', 'Application Server Wizards', 'Server Properties', 'Tools', and 'Problem Determination'. Arrows point to the 'View Logs' link in the navigation bar and the 'View Logs' link in the sidebar.

Log file name	File size
/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/trace.log	6030
/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/messages.log	14160
/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/stopServer-console.log	0
/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/qlwiesvr.log	2772
/QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/Admin5/logs/http_access.log	1978

Figure 5. View logs page

Some files are associated with internal server startup processing. For example, `jobname.txt` and `lwipid.txt`. From the perspective of a programmer or administrator, there are three primary log files for an integrated application server:

- `console.log` - containing the redirected standard output and standard error from the JVM process. This console output is intended for direct human consumption. The console output contains major events and errors. The console output always contains messages that are written directly by the JVM process, such as the output generated by the JVM property `-verbose:gc`.

- `messages.log` - containing all messages except server trace messages that are written or captured by the logging component. All messages that are written to this file contain additional information such as the message time stamp and the ID of the thread that wrote the message. This file does not contain messages that are written directly by the JVM process.
- `trace.log` - containing all messages that are written or captured by the product. This file is created only if you enable the advanced server tracing that is discussed in “Configuring server tracing” on page 21. This file does not contain messages that are written directly by the JVM process.

Figure 6 on page 20 shows sample output written to `messages.log` from a server starting.

```

*****
product = WebSphere Application Server 23.0.0.3 (wlp-1.0.75.c1230320230319-1900)
wlp.install.dir = /QIBM/ProdData/OS/ApplicationServer/runtime/wlp/
server.config.dir = /QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/admin5/
java.home = /QOpenSys/QIBM/ProdData/JavaVM/jdk80/64bit/jre
java.version = 1.8.0_351
java.runtime = Java(TM) SE Runtime Environment (8.0.7.20 - pap6480sr7fp20-20221020_01(SR7 FP20))
os = OS/400 (V7R3M0; ppc64) (en_US)
process = 15913@UT30P44.RCH.STGLABS.IBM.COM
Classpath = /QIBM/ProdData/OS/ApplicationServer/runtime/wlp/bin/
tools/ws-server.jar:/QIBM/ProdData/OS/ApplicationServer/runtime/wlp/bin/tools/ws-javaagent.jar:/
QIBM/ProdData/OS/ApplicationServer/runtime/wlp/bin/tools/ws-javaagent.jar
Java Library path = /QSYS.LIB:/QSYS.LIB/QHTTSPVR.LIB
Suppressed message ids: [CWWKS4001I]
*****
[4/14/23 13:54:11:457 CDT] 00000001
com.ibm.ws.kernel.launch.internal.FrameworkManager          A CWWKE0001I: The server admin5
has been launched.
[4/14/23 13:54:15:847 CDT] 00000025
com.ibm.ws.config.xml.internal.XMLConfigParser              A CWWKG0028A: Processing
included configuration resource: /QIBM/UserData/OS/AdminInst/admin5/wlp/usr/servers/admin5/
resources/security/admin-cust.xml
[4/14/23 13:54:15:871 CDT] 00000025
com.ibm.ws.config.xml.internal.XMLConfigParser              A CWWKG0028A: Processing included
configuration resource: /QIBM/ProdData/OS/RSEAPI/IBMiRSEAPI.xml
[4/14/23 13:54:25:114 CDT] 00000042
com.ibm.ws.http.internal.VirtualHostImpl                   A CWWKT0016I: Web application
available (default_host): http://ut30p44:2011/rseapi/
.
.
[4/14/23 13:54:25:742 CDT] 00000031
com.ibm.ws.kernel.feature.internal.FeatureManager          A CWWKF0012I: The server installed
the following features: [appSecurity-2.0, distributedMap-1.0, e1-3.0, jaxb-2.2, jaxrs-2.0,
jaxrsClient-2.0, jdbc-4.1, jndi-1.0, jsf-2.2, json-1.0, jsp-2.3, mpConfig-1.2, mpOpenAPI-1.1,
servlet-3.1, ssl-1.0, transportSecurity-1.0].
[4/14/23 13:54:25:742 CDT] 00000031
com.ibm.ws.kernel.feature.internal.FeatureManager          I CWWKF0008I: Feature update
completed in 9.334 seconds.
[4/14/23 13:54:25:742 CDT] 00000031
com.ibm.ws.kernel.feature.internal.FeatureManager          A CWWKF0011I: The admin5 server is
ready to run a smarter planet. The admin5 server started in 15.232 seconds.
[4/14/23 13:54:51:839 CDT] 00000031
om.ibm.ws.security.registry.internal.UserRegistryServiceImpl E CWWKS3005E: A configuration
exception has occurred. No UserRegistry implementation service is available. Ensure that you
have a user registry configured.
[4/14/23 13:54:51:888 CDT] 00000031
com.ibm.ws.logging.internal.impl.IncidentImpl              I FFDC1015I: An FFDC Incident
has been created: "com.ibm.ws.security.registry.RegistryException: CWWKS3005E: A
configuration exception has occurred. No UserRegistry implementation service
is available. Ensure that you have a user registry configured.
com.ibm.ws.security.authentication.jaas.modules.UsernameAndPasswordLoginModule 116" at
ffdc_23.04.14.13.54.51.0.log
[4/14/23 13:54:52:102 CDT]
00000031 .ibm.ws.jaxrs.2.0.common:1.0.75.c1230320230319-1900(id=123)] I Setting the server's
publish address to be /api/
[4/14/23 13:54:52:271 CDT] 00000031
com.ibm.ws.webcontainer.servlet                            I SRVE0242I: [IBMiRSEAPI] [/
rseapi] [com.ibm.rse.rest.api.jaxrs.RSEResourceConfig]: Initialization successful.
[4/14/23 13:54:52:382 CDT] 00000048
IBMiRSEAPI                                                I Standard logger starting.
[4/14/23 13:54:52:670 CDT] 00000048
IBMiRSEAPI                                                A login: User 'amra' authenticated.

```

Figure 6. Sample output of `messages.log`

The log shows the server startup procedure. First, the kernel starts, then the feature manager initializes and reads the configuration files. The server is configured to listen on given ports. The RSE API web application is started. The log also shows an audit record for a client login request.

**Note:** The log records CWWKS3005E and FFDC1015I can be ignored.

Enabling trace produces lots of information and negatively affects the performance of RSE API, so you do not want to enable traces in a production environment unless you really need to.

## Configuring server tracing

The integrated application server can be configured to gather debug information for the server runtime.

**Note:** Server runtime tracing is typically asked for by IBM service to debug a problem in the server.

You can modify the server runtime tracing level by clicking on the **Server Logging** tab as shown in [Figure 7](#) on page 21.

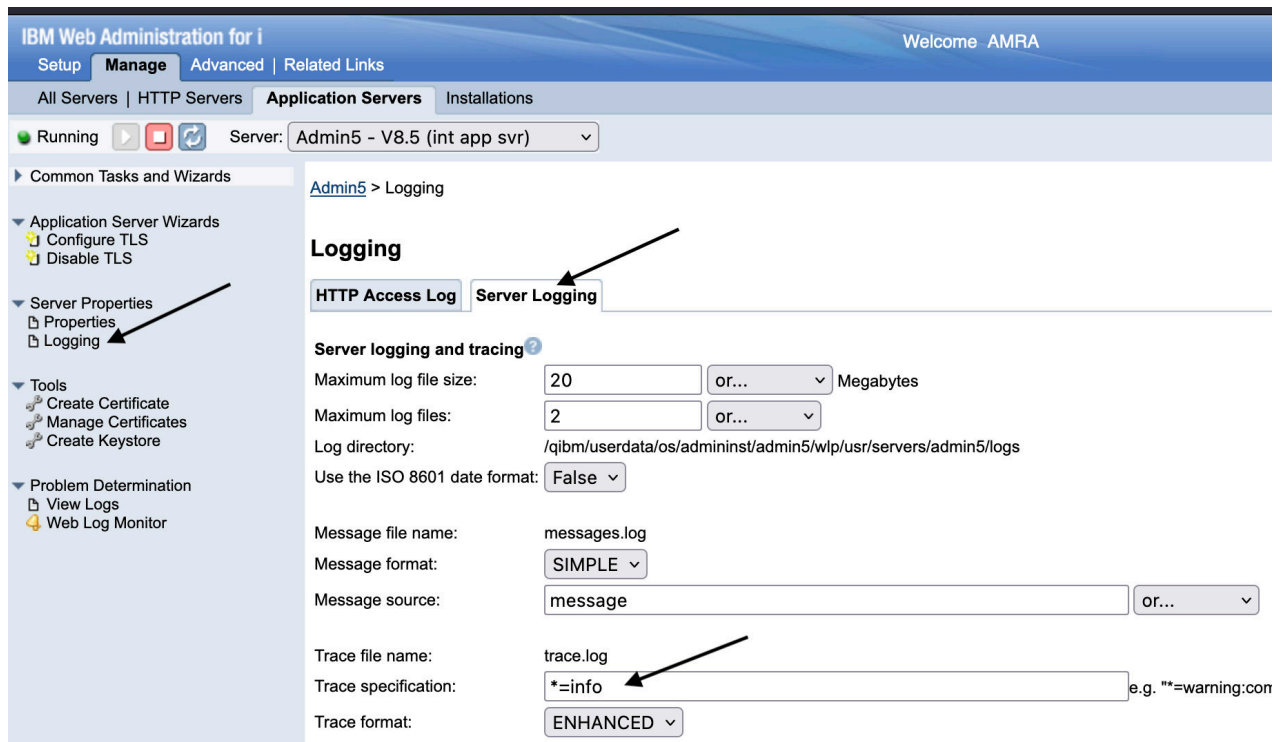


Figure 7. Server Logging

The format of the trace specification is:

```
<component> = <level>
```

where <component> is the component for which to set a log detail level, and <level> is one of the valid logger levels shown in [Table 3](#) on page 22. Separate multiple trace specifications with colons (:).

Components correspond to Java packages and classes, or to collections of Java packages. Use an asterisk (\*) as a wildcard to indicate components that include all the classes in all the packages that are contained by the specified component. For example:

**\***

Specifies all traceable code that is running in the server, including the product system code and customer code.

**com.ibm.ws.\***

Specifies all classes with the package name beginning with com.ibm.ws.

## **com.ibm.ws.classloader.JarClassLoader**

Specifies the JarClassLoader class only.

The following table shows valid logging levels:

<i>Table 3. Valid logging levels.</i>	
<b>Logging level</b>	<b>Description</b>
off	Logging is turned off.
fatal	Task cannot continue and component, application, and server cannot function.
severe	Task cannot continue but component, application, and server can still function. This level can also indicate an impending unrecoverable error.
warning	Potential error or impending error. This level can also indicate a progressive failure (for example, the potential leaking of resources).
audit	Significant event that affects server state or resources.
info	General information that outlines overall task progress.
config	Configuration change or status.
detail	General information that details subtask progress.
fine	General trace information plus method entry, exit, and return values.
finer	Detailed trace information.
finest	A more detailed trace that includes all the detail that is needed to debug problems.
all	All events are logged, includes custom levels, and can provide a more detailed trace than finest.

## **HTTP access logging**

An HTTP access log contains a record of all inbound client requests handled by HTTP endpoints. By default, HTTP access logging is disabled. You can enable access logging in the integrated application server by navigating to the **HTTP Access Log** tab as shown in [Figure 8 on page 23](#):

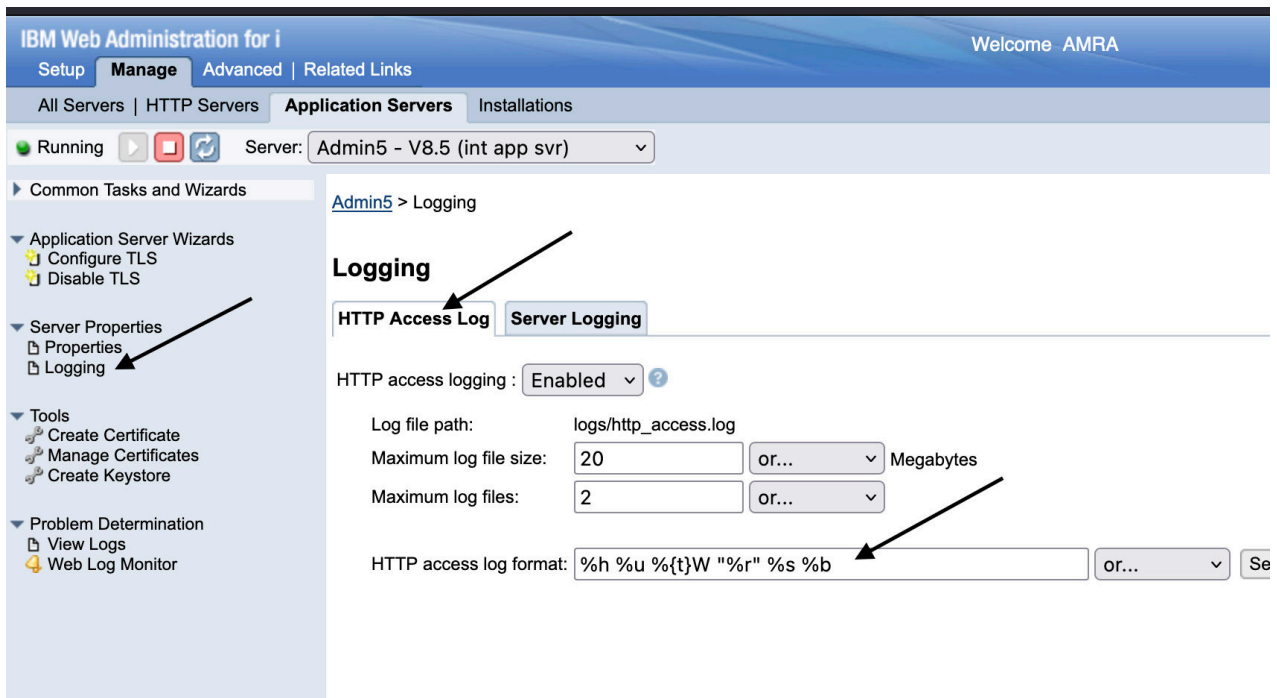


Figure 8. HTTP access log

The **HTTP access log format** is used when logging client access information. The value for this property is a space-separated list of options. The order that you specify the options determines the format of this information in the log. The default value is as follows:

```
%h %u %i{t}W "%I" %s %b
```

Each option can be enclosed in quotation marks, but the quotation marks are not required. Unless otherwise noted, a value of '-' is printed for an option if the requested information cannot be obtained for that option. The following list indicates the available options and the information that is printed if that option is specified as part of the value specified for this property.

**%a**

Remote address.

**%A**

Local IP address.

**%b**

Response size in bytes excluding headers.

**%B**

Response size in bytes excluding headers. The number 0 is printed instead of dash (-) if no value is found.

**%{CookieName}C or %C**

The request cookie specified within the brackets, or if the brackets are not included, prints all request cookies.

**%D**

The elapsed time of the request - millisecond accuracy, microsecond precision.

**%h**

Remote host.

**%{HeaderName}i**

The header name specified within the brackets from the request.

**%m**

Request method.

**%{HeaderName}o**

The header name specified within the brackets from the response.

**%q**

Output the query string with any password escaped.

**%r**

First line of the request.

**%{R}W**

Service time of the request from the moment the request is received until the first set of bytes of the response is sent - millisecond accuracy, microsecond precision. The %{R}W option is often a good approximation for application response time (as compared to %D, which is end-to-end response time including client and network).

**%s**

Status code.

**%t**

NCSA format of the start time of the request.

**%{t}W**

The current time when the message to the access log is queued to be logged in normal NCSA format.

**%u**

Remote user according to the WebSphere Application Server specific \$WSRU header.

**%U**

URL Path, not including the query string.

## JVM dumps

The integrated application server runs in a Java Virtual Machine (JVM). As such, you have the ability, with the proper authority, to diagnose problems at the JVM level, such as hung threads, deadlocks, excessive processing, excessive memory consumption, memory leaks, and defects in the virtual machine.

The most straightforward way to look at various aspects of the server is by using the Work with Java Virtual Machine (WRKJVMJOB) CL command. The following information or functionality is available for IBM Technology for Java JVM jobs:

- The arguments and options with which the JVM was started.
- Environment variables for both ILE and PASE.
- Java locks being blocked, held, and waiting on.
- Garbage collection information.
- The properties with which the JVM was started.
- The properties with which the JVM is running.
- The list of threads associated with the JVM.
- The partially completed job log for the JVM job.
- The ability to generate JVM (System, Heap, and Java) dumps.
- The ability to enable and disable verbose garbage collection.

To learn more about the WRKJVMJOB command, search on the command in the [IBM Documentation](#).

---

## RSE API reference

This part of the document describes various API methods available to users. Things to note:

- In general, all requests and responses are in JSON format.
- RSE API uses conventional HTTP response codes to indicate the success or failure of an API request. In general, codes in the 2xx range indicate success. Codes in the 4xx range indicate an error that failed given the information that is provided (for example, a required parameter was omitted). Codes in the 5xx range indicate an internal error with RSE API. If an error is encountered, the following JSON response is returned (the following is an example of an error):

```
{
  "title": "BAD_REQUEST",
  "status": 400,
  "detail": "Path name not set.",
  "method": "GET",
  "instance": "/rseapi/api/v1/ifs/list",
  "timestamp": "2022-02-20T14:51:58.252Z"
}
```

where:

- *title* is a string representing the HTTP status code description.
  - *status* is an integer representing the HTTP status code.
  - *detail* is a string that provides more details about the error.
  - *method* is a string that represents the HTTP method that was used in the request.
  - *instance* is a string that represents the URI that was used in the request.
  - *timestamp* is a string that represents a timestamp of when the error occurred in ISO-8601 format.
- In the API documentation, the **Parameters** section for each API lists the parameters and an indication of the source of the parameter value:
    - (*header*) indicates that the parameter value is obtained from an HTTP header.
    - (*query*) indicates that the parameter value is obtained from the query string parameter in the URL.
    - (*path*) indicates that the parameter value is obtained from a path component of the URL path. The path variable will be shown with a variable name that is encapsulated with a leading and trailing brace. For example, in the following IFS API path, `/api/v1/ifs/{path}/info`, the component in the path is designated by `{path}`.





---

# API methods: Administration Services

Administration Services provide APIs that give information about RSE API and the runtime environment of the RSE API server. To use the APIs, the authenticated user must authenticate to localhost and have the administrator role or have \*ALLOBJ special authority.

---

## GET /api/v1/admin/settings

---

Get the general settings being used for RSE API.

The settings are global in nature and include settings for tuning, environment, and administrator override categories.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Responses

#### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "adminUsers": ["USER1", "USER2"],
  "includeUsers": ["USER1", "USER2"],
  "excludeUsers": [],
  "maxFileSize": 3072000,
  "maxSessionInactivity": 7200,
  "maxSessionLifetime": 21600,
  "maxSessionUseCount": 1000,
  "maxSessionWaitTime": 300,
  "maxSessions": 100,
  "maxSessionsPerUser": 20,
  "sessionCleanupInterval": 300
}
```

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

# POST /api/v1/admin/settings

---

Set settings for RSE API.

The settings are global in nature and include settings for tuning, environment, and administrator override categories.

## Parameters

### Authorization (header)

The authorization HTTP header.

**type**  
string

## Request body

The settings for RSE API.

**Required:** true

### Media types

**application/json**

#### Schema

[“RSEAPI\\_Settings” on page 78](#)

#### Example

```
{
  "persist": false,
  "adminUsers": ["USER1", "USER2"],
  "includeUsers": ["USER1", "USER2"],
  "excludeUsers": [],
  "maxFileSize": 3072000,
  "maxSessionInactivity": 7200,
  "maxSessionLifetime": 21600,
  "maxSessionUseCount": 1000,
  "maxSessionWaitTime": 300,
  "maxSessions": 100,
  "maxSessionsPerUser": 20,
  "sessionCleanupInterval": 300
}
```

## Responses

### 204

Successful request, no content.

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/admin/sessions

---

Get information about sessions.

Get information about sessions. The information that is returned applies to active sessions on the server. Active sessions may include sessions that are expired but have not been reclaimed by RSE API.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Responses

#### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "totalSessions": 3,
  "sessions": [
    {
      "userid": "USER1",
      "sessionCount": 1
    },
    {
      "userid": "USER2",
      "sessionCount": 2
    }
  ]
}
```

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## DELETE /api/v1/admin/sessions

---

Delete sessions.

Delete sessions. You can delete all active sessions or only sessions tied to a user ID. Sessions that are deleted are marked as expired.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

#### user (query)

The session(s) to delete. Specify the user ID to delete all sessions created by the user, or the special value of \*ALL to delete all sessions for all users.

**type**  
string

**default**  
\*ALL

### Responses

#### 204

Successful request, no content.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/admin/memory

---

Get information about server memory usage.

Get information about the JVM memory usage of the server running RSE API.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

## Responses

### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "jvmFreeMemory": 17428920,
  "jvmMaxMemory": 4294967296,
  "jvmTotalMemory": 78249984
}
```

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/admin/environment

---

Get information about server environment.

Get information about server environment, such as host, operating system, Java version, and port.

## Parameters

### Authorization (header)

The authorization HTTP header.

#### type

string

## Responses

### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "rseapiBasepath": "rseapi",
  "rseapiHostname": "UT30P44",
  "rseapiPort": 2012,
  "rseapiVersion": "1.0.6",
  "osName": "OS/400",
  "osVersion": "V7R5M0",
}
```

```
} "javaVersion": "1.8.0_351"
```

#### **401**

Unauthorized request was made.

#### **403**

The request is forbidden.

#### **500**

Unable to process the request due to an internal server error.

### **Security Requirements**

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

---

# API methods: CL Command Services

CL Command Services provide APIs for running CL commands.

## PUT /api/v1/cl

---

Run one or more CL commands on the server.

Run one or more CL commands on the server. If a command fails, any messages relating to the error is returned. If the command succeeds, no data is returned. By default, a response payload will be returned if an error occurs that will include the first level message text. If you want the full message details, set the `includeMessageHelpText` property to true. If you want messages returned in all cases, set the `includeMessageOnSuccess` property to true.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Request body

List of CL commands to run. If more than one CL command is to be run, you can indicate whether all commands should be run even if an error is encountered while running a CL command.

### Media types

#### application/json

##### Schema

["RSEAPI\\_CLCommands" on page 75](#)

##### Example

```
{
  "continueOnError": true,
  "includeMessageOnSuccess": true,
  "includeMessageHelpText": false,
  "clCommands": [
    "qsys/crtlib lib1",
    "qsys/crtsrcpf lib1/qrpglesrc"
  ]
}
```

### Responses

#### 200

Command(s) issued successfully, error(s) may have occurred.

### Media types

#### application/json

##### Example

```
{
  "totalIssued": 2,
  "totalSuccesses": 1,
  "totalFailures": 1,
  "commandOutputList": [
    {}
  ]
}
```

```

    "success": false,
    "command": "qsys/crtlib lib1",
    "output": [
      "CPF2111: Library LIB1 already exists. "
    ]
  },
  {
    "success": true,
    "command": "qsys/crtsrcpf lib1/qrpglesrc",
    "output": [
      "CPC7301: File QRPGLESRC created in library LIB1. "
    ]
  }
]
}

```

#### 204

All command(s) issued successfully, no content.

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/cl/{commandname}

---

Retrieves the command definition for the specified CL command.

Retrieves the command definition for the specified CL command. The command definition is returned as an XML document. The generated command information XML source is called Command Definition Markup Language or CDML. See the Document Type Definition (DTD) in /QIBM/XML/DTD/QcdCLCmd.dtd for the definition of the CDML tag language returned by this API.

### Parameters

#### Authorization (header)

**type**  
string

#### library (query)

The library name. Valid values are a specific name, or one of the following special values:

- \*CURLIB - The current library is searched.
- \*LIBL - The library list is searched. This is the default.

**type**  
string

**default**  
\*LIBL



### ignorecase (query)

Boolean indicating whether case should be ignored. The default value is 'true'. If set to 'false', the library and command name will not be uppercased.

**type**  
boolean

**default**  
true

### commandname (path)

The CL command name.

**type**  
string

**required**  
true

## Responses

### 200

Successful request.

#### Media types

##### application/json

##### Example

```
{
  "definition": "<QcdCLCmd DTDVersion=\"1.0\"><Cmd CmdName=\"DSPLIBL\"
CmdLib=\"__LIBL\" CCSID=\"37\" HlpPnlGrp=\"QHLCMD\" HlpPnlGrpLib=\"__LIBL\"
HlpID=\"DSPLIBL\" MaxPos=\"1\" Prompt=\"Display Library List\" MsgF=\"QCPFMSG\"
MsgFLib=\"__LIBL\" ExecBatch=\"YES\" ChgCmdExit=\"NO\" RtvCmdExit=\"NO\"><Parm
Kwd=\"OUTPUT\" PosNbr=\"1\" KeyParm=\"NO\" Type=\"CHAR\" Min=\"0\" Max=\"1\"
Prompt=\"Output\" Len=\"1\" Rstd=\"YES\" Dft=\"*\" AlwUnprt=\"YES\"
AlwVar=\"YES\" Expr=\"YES\" Full=\"NO\" DspInput=\"YES\" Choice=\"*, *PRINT\"
><SpcVal><Value Val=\"*\" MapTo=\"*\"/><Value Val=\"*PRINT\" MapTo=\"L\"/></
SpcVal></Parm></Cmd></QcdCLCmd>"
}
```

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 404

The specified resource was not found.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)



---

# API methods: IFS Services

Integrated File System (IFS) Services provide APIs for accessing objects in a way that is like personal computer and UNIX operating systems. This includes listing objects in directories, reading from files, and writing to files.

---

## GET /api/v1/ifs/list

Gets a list of objects in the specified path.

Gets a list of objects in the specified path. The information returned includes the name, whether object is a directory, the description, and the object subtype. For objects that are not in the QSYS.LIB file system, any part of the path may contain an asterisk (\*), which is a wildcard that means zero or more instances of any character. For example, a path of '/tmp/am\*1.txt' will return all objects in directory '/tmp' that have names that begin with 'am' and end with '1.txt'. For QSYS.LIB objects, only generic names may be specified in any part of the path. A generic name is a character string that contains one or more characters followed by an asterisk. For example, '/qsys.lib/am\*.lib' will return all libraries that have names that start with 'am'. Another example is '/qsys.lib/am\*.\*', which would return all objects that start with 'am'. This would be equivalent to specifying a path of '/qsys.lib/am\*'.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

#### path (query)

The working directory. For example, /u/IBM/test

**type**  
string

**required**  
true

#### subtype (query)

Subtype of objects to return. Valid values include a specific object type (\*LIB, \*FILE, \*PGM, \*OUTQ, etc.) or \*ALL. Note that many file system objects do not have a subtype. For example, any Root, QOpenSys or UDFS object.

**type**  
string

#### includehidden (query)

Whether to show hidden files.

**type**  
boolean

**default**  
false

### Responses

#### 200

Successful request.

## Media types

### application/json

#### Example

```
{
  "objects": [
    {
      "path": "/QSYS.LIB/USER1.LIB/QCLSRC.FILE",
      "description": "\"test\"",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/QCSRC.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/QRPGLESRC.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/QSQDSRC.FILE",
      "description": "SQL PROCEDURES",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/QSRVSRV.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-DTA"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/QWOBJ.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/QAUDJR0043.JRNRCV",
      "description": "",
      "isDir": false,
      "subType": ""
    }
  ]
}
```

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 404

The specified resource was not found.

#### 500

Unable to process the request due to an internal server error.

## Security Requirements

## GET /api/v1/ifs/{path}

---

Get the content of a file.

Get the content of a file. The content is returned in a JSON object.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

#### ETag (header)

Whether to return checksum for the file.

**type**  
boolean

#### path (path)

Path to file for which the data is to be read.

**type**  
string

**required**  
true

### Responses

#### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "ccsid": 819,
  "content": "Hello world\n"
}
```

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 404

The specified resource was not found.

#### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## PUT /api/v1/ifs/{path}

---

Write a string to a file.

Write a string to a file. The file must exist and its contents will be replaced by the string specified in the request.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

#### If-Match (header)

If the If-Match HTTP header is passed, RSE API will check to see if the Etag (MD5 hash of the object content) matches the provided Etag value. If this value matches, the operation will proceed. If the match fails, the system will return a 412 (Precondition Failed) error.

**type**  
string

#### path (path)

Path to file in which the data will be written.

**type**  
string

**required**  
true

### Request body

File content.

**Required:** true

#### Media types

##### application/json

###### Schema

["RSEAPI\\_FileContent" on page 80](#)

###### Example

```
{
  "content": "some data that will be written to file."
}
```

##### text/plain

###### Schema

###### Example

```
some data that will be written to file.
```

## Responses

### 204

Successful request, no content.

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 404

The specified resource was not found.

### 412

Precondition failed.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/ifs/{path}/info

---

Returns information about the object referenced by the path.

Returns information about the object referenced by the path. The information returned includes the name, whether object is a directory, the description, the object subtype, CCSID, size, last modified timestamp, and object subtype.

## Parameters

### Authorization (header)

The authorization HTTP header.

**type**

string

### path (path)

Path to object for which information is to be returned.

**type**

string

**required**

true

## Responses

### 200

Successful request.

## Media types

### application/json

#### Example

```
{
  "path": "/qsys.lib/user1.lib",
  "description": "user1's lib",
  "isDir": true,
  "subType": "PROD",
  "owner": "USER1",
  "ccsid": 37,
  "lastModified": 1680559633,
  "size": 401408,
  "recordLength": -1,
  "numberOfRecords": -1
}
```

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 404

The specified resource was not found.

#### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)



---

# API methods: QSYS Services

QSYS Services provide APIs for accessing QSYS objects.

---

## GET /api/v1/qsys/search/{objectName}

---

Returns a list of QSYS.LIB objects that match the search criteria.

Returns a list of QSYS.LIB objects that match the search criteria. The filter is the object name, and may be a value of \*ALL, in which case all objects that match the search criteria is returned, or a generic name. A generic name is a character string that contains one or more characters followed by an asterisk (\*). If a generic name is specified, all objects that have names with the same prefix as the generic name are returned.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

#### objectLibrary (query)

The library or set of libraries that are searched for objects. Valid values are a specific name, or one of the following special values:

- \*ALL - All libraries are searched.
- \*ALLUSR - All user libraries are searched.
- \*CURLIB - The current library is searched.
- \*LIBL - The library list is searched.
- \*USRLIBL - The user portion of the library list is searched.

**type**  
string

**default**  
\*USRLIBL

#### objectType (query)

The type of object to search. Valid values include: \*FILE, \*PGM, \*LIB, etc., or \*ALL.

**type**  
string

**default**  
\*ALL

#### objectSubtype (query)

Object subtype. For example, PF-SRC, PF-DTA, SAVF, etc., or \*ALL. Note that not all objects have subtypes.

**type**  
string

**default**  
\*ALL

### memberName (query)

The member name to match for objects of type PF-SRC or PF-DTA. Valid values are a specific name, or an extended generic name where the asterisk (\*) may be placed in any part of the name, or \*ALL. Note that members are not searched for unless the object subtype that starts with the prefix PF. For example, PF, or PF-SRC.

**type**  
string

**default**

### memberType (query)

The member type to match for members of objects of type PF-SRC or PF-DTA. Valid values are a specific name, such as C, CBLLE, RPGLE, SQLRPGLE, etc., or \*ALL.

**type**  
string

**default**  
\*ALL

### objectName (path)

The object name. Valid values are a specific name, a generic name (for example, AM\*), or one of the following special values:

- \*ALL - All object names are searched.
- \*ALLUSR - All objects that are libraries in QSYS or the library list are searched. The object library must either be \*ALL, \*LIBL or QSYS. The object type must be \*LIB. A list of user libraries is returned.

**type**  
string

**required**  
true

## Responses

### 200

Successful request.

### Media types

**application/json**

#### Example

```
{
  "objects": [
    {
      "path": "/QSYS.LIB/USER1.LIB/AXISLIBS.FILE",
      "description": "",
      "isDir": false,
      "subType": "SAVF"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/DEALER.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-DTA"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/FLGHT400.FILE",
      "description": "",
      "isDir": false,
      "subType": "SAVF"
    },
    {
      "path": "/QSYS.LIB/USER1.LIB/IWSDB1.FILE",
      "description": ""
    }
  ]
}
```

```
    "isDir": true,  
    "subType": "PF-DTA"  
  },  
  {  
    "path": "/QSYS.LIB/USER1.LIB/QCLSRC.FILE",  
    "description": "\"test\"",  
    "isDir": true,  
    "subType": "PF-SRC"  
  },  
  {  
    "path": "/QSYS.LIB/USER1.LIB/QCSRC.FILE",  
    "description": "",  
    "isDir": true,  
    "subType": "PF-SRC"  
  }  
]  
}
```

#### **400**

Bad request.

#### **401**

Unauthorized request was made.

#### **403**

The request is forbidden.

#### **404**

The specified resource was not found.

#### **500**

Unable to process the request due to an internal server error.

### **Security Requirements**

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)



---

# API methods: Security Services

Security Services provide APIs relating to security, such as the management of digital certificates and the retrieval of TLS system information.

All the digital certificate management APIs require the Digital Certificate Manager, option 34 of the IBM i licensed program (5770-SS1), be installed. In addition, the authenticated user must have the \*ALLOBJ and \*SECADM special authorities.

---

## POST /api/v1/security/dcm/appdef/associate

Associate digital certificates to an application definition.

Associate digital certificates to an application definition. A maximum of 4 certificates can be specified.

On successful completion, the specified certificates will replace any pre-existing certificates associated with the application definition.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Request body

The API properties required to assign digital certificates to an application definition.

**Required:** true

#### Media types

**application/json**

##### Schema

[“RSEAPI\\_DCMCertAppDefAssociateRequest” on page 76](#)

##### Example

```
{
  "appDefinitionID": "myappdef",
  "certAliases": ["mylabel1", "mylabel2"]
}
```

### Responses

#### 204

Request successful, no content.

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

## 404

The specified resource was not found.

## 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/appdef/disassociate

---

Disassociate digital certificates from an application definition.

Disassociate digital certificates from an application definition. All certificates associated with the application definition will be disassociated.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Request body

The API properties required to disassociate digital certificates from an application definition.

**Required:** true

#### Media types

**application/json**

#### Schema

[“RSEAPI\\_DCMCertAppDefDisassociateRequest” on page 83](#)

#### Example

```
{
  "appDefinitionID": "myappdef"
}
```

### Responses

#### 204

Request successful, no content.

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

## 404

The specified resource was not found.

## 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/security/dcm/appdef/list

---

List application definitions.

Retrieve a list of application definitions. You can filter what is returned by application definition ID and application type. When filtering by application definition ID, you can specify a generic ID. A generic ID is a character string that contains one or more characters followed by an asterisk (\*). If a generic ID is specified, all application definition IDs that have an ID with the same prefix as the generic ID are returned.

## Parameters

### Authorization (header)

The authorization HTTP header.

**type**  
string

### idFilter (query)

Application definition ID filter.

**type**  
string

### typeFilter (query)

Application type filter. Possible values: SERVER, CLIENT, SERVER\_CLIENT, OBJECT\_SIGNING. If not specified, all server and client application definitions are returned.

**type**  
string

## Responses

### 200

Successful request.

### Media types

**application/json**

#### Example

```
{
  "appDefinitions": [
    {
      "appDefinitionID": "QIBM_OS400_QRW_SVR_DDM_DRDA",
      "appType": "SERVER",
      "description": "IBM i DDM/DRDA Server - TCP/IP",
      "certAliases": [ ]
    }
  ]
}
```

```
} ]
```

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 404

The specified resource was not found.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/appdef/trust

---

Add certificate authority (CA) digital certificate to the application definition CA trust list.

Add a CA certificate to the application definition CA trust list.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Request body

The API properties required to add an CA to the application definition CA trust list.

**Required:** true

#### Media types

**application/json**

##### Schema

["RSEAPI\\_DCMCertAppDefTrustRequest" on page 76](#)

##### Example

```
{  
  "appDefinitionID": "myappdef",  
  "certAlias": "mylabel1"  
}
```

### Responses

#### 204

Request successful, no content.



**400**

Bad request.

**401**

Unauthorized request was made.

**403**

The request is forbidden.

**404**

The specified resource was not found.

**500**

Unable to process the request due to an internal server error.

**Security Requirements**

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/appdef/untrust

---

Remove a certificate authority (CA) digital certificate from the application definition CA trust list.

Remove a certificate authority (CA) digital certificate from the application definition CA trust list.

**Parameters****Authorization (header)**

The authorization HTTP header.

**type**  
string

**Request body**

The API properties required to remove a CA digital certificate from an application definition CA trust list.

**Required:** true

**Media types**

**application/json**

**Schema**

["RSEAPI\\_DCMCertAppDefTrustRequest"](#) on page 76

**Example**

```
{
  "appDefinitionID": "myappdef",
  "certAlias": "mylabel1"
}
```

**Responses****204**

Request successful, no content.

**400**

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 404

The specified resource was not found.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/cert/delete

---

Delete a digital certificate.

Delete a digital certificate.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Request body

The API properties required to delete a digital certificate.

**Required:** true

#### Media types

**application/json**

#### Schema

[“RSEAPI\\_DCMCertRequest” on page 81](#)

#### Example

```
{
  "certStoreType": "CMS",
  "certStorePath": "*SYSTEM",
  "certStorePassword": "passw0rd",
  "certAlias": "mylabel"
}
```

### Responses

#### 204

Request successful, no content.

#### 400

Bad request.

## 401

Unauthorized request was made.

## 403

The request is forbidden.

## 404

The specified resource was not found.

## 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/cert/export

---

Export a digital certificate.

Export a digital certificate. Only server/client and CA certificates can be exported. Certificates can be exported in the DER, PEM, or PKCS12 formats. A server/client or CA certificate that is to include the private key must be exported in the PKCS12 format. CA certificates without private keys cannot be exported in the PKCS12 format.

When exporting certificates in the PKCS12 format, a password for the exported certificate must be specified.

The certificate data in the response will be encoded in Base64, even for certificate data returned in the PEM format.

## Parameters

### Authorization (header)

The authorization HTTP header.

**type**  
string

## Request body

The API properties required to export a digital certificate.

**Required:** true

### Media types

**application/json**

#### Schema

[“RSEAPI\\_DCMCertExportRequest” on page 85](#)

#### Example

```
{
  "certStoreType": "CMS",
  "certStorePath": "*SYSTEM",
  "certStorePassword": "passw0rd",
  "certFormat": "PKCS12",
  "certAlias": "mylabel",
  "certDataPassword": "myPassw0rd"
}
```

## Responses

### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "certFormat": "PKCS12",
  "certData": "BASE64-BLOB"
}
```

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 404

The specified resource was not found.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/cert/import

---

Import a digital certificate.

Import a digital certificate. Only server/client or CA certificates can be imported. A certificate can be imported in the following formats: PKCS12, DER, or PEM. If the certificate to be imported includes a private key, then the PKCS12 format must be used.

If importing a CA certificate and the certificate includes a private key, the PKCS12 format must be used and the certificate type must be set to SERVER\_CLIENT. When importing CA certificate that do not contain a private key, the PEM or DER format must be used.

The certificate data in the request must be encoded in Base64, which includes certificate data in the PEM format.

## Parameters

### Authorization (header)

The authorization HTTP header.

#### type

string

## Request body

The API properties required to import a digital certificate.

**Required:** true

### Media types

**application/json**

#### Schema

[“RSEAPI\\_DCMCertImportRequest” on page 84](#)

#### Example

```
{
  "certStoreType": "CMS",
  "certStorePath": "*SYSTEM",
  "certStorePassword": "passw0rd",
  "certType": "SERVER_CLIENT",
  "certFormat": "PKCS12",
  "certAlias": "mylabel",
  "certData": "BASE64-BLOB",
  "certDataPassword": "myPassw0rd"
}
```

## Responses

### 204

Request successful, no content.

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 404

The specified resource was not found.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/cert/info

---

Get detailed certificate information.

Get detailed certificate information, such as subject, issuer, subject alternative names, and serial number.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

## Request body

The API properties required to get detailed information about a digital certificate.

**Required:** true

### Media types

**application/json**

#### Schema

["RSEAPI\\_DCMCertRequest" on page 81](#)

#### Example

```
{
  "certStoreType": "CMS",
  "certStorePath": "*SYSTEM",
  "certStorePassword": "passw0rd",
  "certAlias": "mylabel"
}
```

## Responses

### 200

Successful request.

### Media types

**application/json**

#### Example

```
{
  "alias": "UNIQUE-SAN",
  "trusted": true,
  "subject": "C=US,SP=Minnesota,O=IBM,CN=Unique",
  "issuer": "C=US,SP=Any,O=IBM Web Administration for i,CN=mysystem_CERTIFICATE_AUTHORITY",
  "keyAlgorithm": "ECDSA",
  "keySize": 256,
  "hasPrivateKey": true,
  "signatureAlgorithm": "RSA_SHA256",
  "keyStorageLocation": "SOFTWARE",
  "serialNumber": "6526CFBC0601B8",
  "effectiveDate": "2024-05-02T23:00:00-05:00",
  "expirationDate": "2026-05-02T22:59:59-05:00",
  "subjectAlternativeNames": [
    "booboo.ibm.com",
    "9.9.9.9"
  ],
  "keyUsages": [
    "DIGITAL_SIGNATURE",
    "NONREPUDIATION",
    "KEY_ENCIPHERMENT",
    "KEY_AGREEMENT"
  ]
}
```

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

## 404

The specified resource was not found.

## 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/cert/list

---

Retrieve a list of certificates in a certificate store.

Retrieve a list of certificates in a certificate store. You can filter what is returned by alias, certificate type, days until expiration, and whether to include expired certificates. When filtering by alias, you can specify a generic alias for alias. A generic alias is a character string that contains one or more characters followed by an asterisk (\*). If a generic alias is specified, all certificates that have an alias with the same prefix as the generic alias are returned. Note that if the daysUntilExpiration filter is specified, CSR certificates will not be returned in the response since CSR certificates do not expire.

## Parameters

### Authorization (header)

The authorization HTTP header.

**type**  
string

## Request body

The API properties required to list certificates in a certificate store.

**Required:** true

### Media types

**application/json**

#### Schema

[“RSEAPI\\_DCMCertListRequest” on page 81](#)

#### Example

```
{
  "certStoreType": "CMS",
  "certStorePath": "*SYSTEM",
  "certStorePassword": "passw0rd",
  "filters": {
    "certAlias": "*",
    "certTypes": ["SERVER_CLIENT", "CA", "CSR"],
    "daysUntilExpiration": 5000,
    "excludeExpired": false
  }
}
```

## Responses

### 200

Successful request.

## Media types

### application/json

#### Example

```
{
  "certificates": [
    {
      "certAlias": "GLOBAL-MULTISAN",
      "commonName": "myserver.ibm.com",
      "type": "SERVER_CLIENT",
      "daysBeforeExpiration": 1831,
      "keyAlgorithm": "ECDSA",
      "keySize": 256,
      "keyStorageLocation": "SOFTWARE",
      "signatureAlgorithm": "ECDSA_SHA256"
    },
    {
      "certAlias": "LOCAL_CERTIFICATE_AUTHORITY_106F947K(5)",
      "commonName": "Local CA for myserver on July 17",
      "type": "CA",
      "daysBeforeExpiration": 6807,
      "keyAlgorithm": "ECDSA",
      "keySize": 256,
      "signatureAlgorithm": "ECDSA_SHA256"
    }
  ]
}
```

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 404

The specified resource was not found.

#### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## POST /api/v1/security/dcm/certstore/changepassword

---

Change digital certificate store password.

Change digital certificate store password. For system certificate stores of type CMS, if the current password is omitted, the system stash file will be used.

### Parameters

#### Authorization (header)

The authorization HTTP header.

#### type

string



## Request body

The API properties required to change a digital certificate store password.

**Required:** true

### Media types

**application/json**

#### Schema

["RSEAPI\\_DCMCertStoreChangePasswordRequest" on page 77](#)

#### Example

```
{
  "certStoreType": "CMS",
  "certStorePath": "*SYSTEM",
  "certStorePassword": null,
  "certStorePasswordNew": "myNewPassw0rd",
  "daysToExpiration": 0
}
```

## Responses

### 204

Request successful, no content.

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/security/tls

---

Retrieve system transport layer security (TLS) attributes.

The API retrieves TLS attributes for the system. The system level settings are based on TLS System Values and System Service Tools (SST) Advanced Analysis command TLSCONFIG that allows viewing or altering of system-wide system TLS default properties.

## Parameters

### Authorization (header)

The authorization HTTP header.

#### type

string

## Responses

200

Successful request.

### Media types

application/json

#### Example

```
{
  "supportedProtocols": [
    "TLSv1.3",
    "TLSv1.2"
  ],
  "eligibleDefaultProtocols": [
    "TLSv1.3",
    "TLSv1.2"
  ],
  "defaultProtocols": [
    "TLSv1.3",
    "TLSv1.2"
  ],
  "supportedCipherSuites": [
    "AES_128_GCM_SHA256",
    "AES_256_GCM_SHA384",
    "CHACHA20_POLY1305_SHA256",
    "ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256",
    "ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256"
  ],
  "eligibleDefaultCipherSuites": [
    "AES_128_GCM_SHA256",
    "AES_256_GCM_SHA384",
    "CHACHA20_POLY1305_SHA256",
    "ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256",
    "ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256"
  ],
  "defaultCipherSuites": [
    "AES_128_GCM_SHA256",
    "AES_256_GCM_SHA384",
    "CHACHA20_POLY1305_SHA256",
    "ECDHE_ECDSA_WITH_AES_128_GCM_SHA256",
    "ECDHE_ECDSA_WITH_AES_256_GCM_SHA384",
    "ECDHE_RSA_WITH_AES_128_GCM_SHA256",
    "ECDHE_RSA_WITH_AES_256_GCM_SHA384",
    "ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256",
    "ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256"
  ],
  "supportedSignatureAlgorithms": [
    "ECDSA_SHA512",
    "ECDSA_SHA384",
    "ECDSA_SHA256",
    "RSA_PSS_SHA512",
    "RSA_PSS_SHA384",
    "RSA_PSS_SHA256",
    "RSA_SHA512",
    "RSA_SHA384",
    "RSA_SHA256"
  ],
  "defaultSignatureAlgorithms": [
    "ECDSA_SHA512",
    "ECDSA_SHA384",
    "ECDSA_SHA256",
    "RSA_PSS_SHA512",
    "RSA_PSS_SHA384",
    "RSA_PSS_SHA256",
    "RSA_SHA512",
    "RSA_SHA384",
    "RSA_SHA256"
  ],
  "supportedSignatureAlgorithmCertificates": [
    "ECDSA_SHA512",
    "ECDSA_SHA384",
    "ECDSA_SHA256",
    "ECDSA_SHA224",
    "ECDSA_SHA1",
    "RSA_PSS_SHA512",

```

```

"RSA_PSS_SHA384",
"RSA_PSS_SHA256",
"RSA_SHA512",
"RSA_SHA384",
"RSA_SHA256",
"RSA_SHA224",
"RSA_SHA1",
"RSA_MD5"
], "defaultSignatureAlgorithmCertificates": [
"ECDSA_SHA512",
"ECDSA_SHA384",
"ECDSA_SHA256",
"RSA_PSS_SHA512",
"RSA_PSS_SHA384",
"RSA_PSS_SHA256",
"RSA_SHA512",
"RSA_SHA384",
"RSA_SHA256"
], "supportedNamedCurves": [
"x25519",
"x448",
"Secp256r1",
"Secp384r1",
"Secp521r1"
], "defaultNamedCurves": [
"Secp256r1",
"Secp384r1",
"x25519",
"Secp521r1",
"x448"
], "defaultMinimumRSAKeySize": 0, "handshakeConnectionCounts": false,
"secureSessionCaching": true, "auditSecureTelnetHandshakes": false}

```

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)

## GET /api/v1/security/tls/stats

Retrieve system transport layer security (TLS) statistics.

The API retrieves TLS statistics. The information returned includes TLS handshake connection counts by protocol type and cipher suite on the system since the last reset for the system. The System Service Tools (SST) Advanced Analysis command TLSCONFIG connectionCounts option identifies the system level setting to enable handshake connection counting.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

## Responses

### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "protocolCounters": {
    "TLSv13": 5,
    "TLSv12": 10,
    "TLSv11": 0,
    "TLSv10": 0,
    "SSLv3": 0
  },
  "cipherSuiteCounters": {
    "AES_128_GCM_SHA256": 0,
    "AES_256_GCM_SHA384": 0,
    "CHACHA20_POLY1305_SHA256": 0,
    "ECDHE_ECDSA_AES_128_GCM_SHA256": 0,
    "ECDHE_ECDSA_AES_256_GCM_SHA384": 0,
    "ECDHE_ECDSA_CHACHA20_POLY1305_SHA256": 0,
    "ECDHE_RSA_AES_128_GCM_SHA256": 0,
    "ECDHE_RSA_AES_256_GCM_SHA384": 0,
    "ECDHE_RSA_CHACHA20_POLY1305_SHA256": 0,
    "RSA_AES_128_GCM_SHA256": 15,
    "RSA_AES_256_GCM_SHA384": 0,
    "ECDHE_ECDSA_AES_128_CBC_SHA256": 0,
    "ECDHE_ECDSA_AES_256_CBC_SHA384": 0,
    "ECDHE_RSA_AES_128_CBC_SHA256": 0,
    "ECDHE_RSA_AES_256_CBC_SHA384": 0,
    "RSA_AES_128_CBC_SHA256": 0,
    "RSA_AES_128_CBC_SHA": 0,
    "RSA_AES_256_CBC_SHA256": 0,
    "RSA_AES_256_CBC_SHA": 0,
    "ECDHE_ECDSA_3DES_EDE_CBC_SHA": 0,
    "ECDHE_RSA_3DES_EDE_CBC_SHA": 0,
    "RSA_3DES_EDE_CBC_SHA": 0,
    "ECDHE_ECDSA_RC4_128_SHA": 0,
    "ECDHE_RSA_RC4_128_SHA": 0,
    "RSA_RC4_128_SHA": 0,
    "RSA_RC4_128_MD5": 0,
    "RSA_DES_CBC_SHA": 0,
    "RSA_EXPORT_RC4_40_MD5": 0,
    "RSA_EXPORT_RC2_CBC_40_MD5": 0,
    "ECDHE_ECDSA_NULL_SHA": 0,
    "ECDHE_RSA_NULL_SHA": 0,
    "RSA_NULL_SHA256": 0,
    "RSA_NULL_SHA": 0,
    "RSA_NULL_MD5": 0
  }
}
```

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 404

The specified resource was not found.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)



---

# API methods: SQL Services

SQL Services provide APIs associated with performing SQL operations.

## PUT /api/v1/sql

---

Run a SQL statement on the server.

Run a SQL statement on the server. If SQL statement fails, any messages relating to the error is returned.

SQL state information is returned only if errors are detected. You have the option of indicating whether the state information is returned on all responses. By default, if a SQL statement is run successfully and there is no result set to return, no data is returned in the response.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Request body

The SQL statement to be run on the server.

**Required:** true

#### Media types

**application/json**

##### Schema

["RSEAPI\\_SQLRequest" on page 83](#)

##### Example

```
{
  "alwaysReturnSQLStateInformation": false,
  "treatWarningsAsErrors": false,
  "sqlStatement": "select * from QIWS.QCUSTCDT"
}
```

### Responses

#### 200

SQL statements(s) issued.

#### Media types

**application/json**

##### Example

```
{
  "resultSet": [
    {
      "CUSNUM": 938472,
      "LSTNAM": "Henning",
      "INIT": "G K",
      "STREET": "4859 Elm Ave",
      "CITY": "Dallas",
      "STATE": "TX",
      "ZIPCOD": 75217,
    }
  ]
}
```

```
    "CDTLMT": 5000,  
    "CHGCOD": 3,  
    "BALDUE": 37,  
    "CDTDUE": 0  
  },  
  "CUSNUM": 839283,  
  "LSTNAM": "Jones",  
  "INIT": "B D",  
  "STREET": "21B NW 135 St",  
  "CITY": "Clay",  
  "STATE": "NY",  
  "ZIPCOD": 13041,  
  "CDTLMT": 400,  
  "CHGCOD": 1,  
  "BALDUE": 100,  
  "CDTDUE": 0  
} ]  
}
```

#### **204**

SQL statement(s) issued successfully, no content.

#### **400**

Bad request.

#### **401**

Unauthorized request was made.

#### **403**

The request is forbidden.

#### **500**

Unable to process the request due to an internal server error.

### **Security Requirements**

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)



---

# API methods: Server Information Services

Server Information Services provide APIs about RSE API.

## GET /api/v1/info/serverdetails

---

Get information about the RSE API.

Get information about the RSE API.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

### Responses

#### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "rseapiBasepath": "rseapi",
  "rseapiHostname": "UT30P44",
  "rseapiPort": 2012,
  "rseapiVersion": "1.0.6"
}
```

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)
- [basicHttpAuthentication](#)



---

# API methods: Session Services

Session Services provide APIs for authenticating a user and managing sessions that are tied to an authenticated user. The user must have a user profile on the IBM i server to be accessed. Once authenticated, a bearer token is returned and must be submitted on requests when invoking protected APIs in an HTTP authorization header.

---

## GET /api/v1/session

---

Get information about the session.

Get information about the session. The information returned includes session settings in addition to information about any host server jobs tied to the session.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

#### envvars (query)

Comma separated list of environment variables to return from the remote command host server job.

**type**  
string

#### maxjoblogrecords (query)

Maximum number of message log records to return from the remote command host server job.

**type**  
integer

**default**  
0

#### joblogfilter (query)

Comma separated list of message IDs. Only remote command host server job log messages that do not match the filter message IDs will be returned.

**type**  
string

### Responses

#### 200

Successful request.

#### Media types

**application/json**

#### Example

```
{
  "sessionInfo": {
    "userID": "user1",
    "host": "localhost",
    "expiration": "2023-04-04T00:56:35Z",
    "creation": "2023-04-03T22:04:50Z",
```

```

    "lastUsed": "2023-04-03T22:56:35Z",
    "domain": "rseapi",
    "expired": false
  },
  "sessionSettings": {
    "libraryList": [],
    "clCommands": [],
    "envVariables": {},
    "sqlDefaultSchema": null,
    "sqlTreatWarningsAsErrors": false,
    "sqlProperties": {
      "XA loosely coupled support": "0",
      "access": "all",
      "auto commit": "true",
      "autocommit exception": "false",
      "bidi implicit reordering": "true",
      "bidi numeric ordering": "false",
      "bidi string type": "5",
      "big decimal": "true",
      "block criteria": "2",
      "block size": "32",
      "character truncation": "true",
      "concurrent access resolution": "2",
      "cursor hold": "true",
      "cursor sensitivity": "asensitive",
      "data compression": "true",
      "data truncation": "true",
      "database name": "",
      "date format": "iso",
      "date separator": "",
      "decfloat rounding mode": "half even",
      "decimal separator": "",
      "driver": "toolbox",
      "errors": "basic",
      "extended dynamic": "false",
      "extended metadata": "false",
      "full open": "false",
      "hold input locators": "true",
      "hold statements": "false",
      "ignore warnings": "01003,0100C,01567",
      "lazy close": "false",
      "libraries": "*LIBL",
      "lob threshold": "32768",
      "maximum blocked input rows": "32000",
      "maximum precision": "31",
      "maximum scale": "31",
      "metadata source": "1",
      "minimum divide scale": "0",
      "naming": "system",
      "numeric range error": "true",
      "package": "",
      "package add": "true",
      "package cache": "false",
      "package ccsid": "13488",
      "package criteria": "default",
      "package error": "warning",
      "package library": "QGPL",
      "portNumber": "0",
      "prefetch": "true",
      "proxy server": "",
      "qaqqinilib": "",
      "query optimize goal": "0",
      "query replace truncated parameter": "",
      "query storage limit": "-1",
      "query timeout mechanism": "qqrytimlmt",
      "remarks": "system",
      "secondary URL": "",
      "server trace": "0",
      "sort": "hex",
      "sort language": "",
      "sort table": "",
      "sort weight": "shared",
      "time format": "iso",
      "time separator": ":",
      "trace": "false",
      "transaction isolation": "read uncommitted",
      "translate binary": "false",
      "translate boolean": "true",
      "translate hex": "character",
      "true autocommit": "false",
      "use block update": "false",
      "variable field compression": "all"
    }
  }
}

```

```

    },
    "sqlStatements": []
  },
  "jobRemoteCommand": {
    "id": "094268/QUSER/QZRCRVS",
    "ccsid": 37,
    "homeDirectory": "/home/USER1",
    "curlib": null,
    "systemLibl": [
      {
        "name": "QSYS",
        "attribute": "PROD",
        "description": "System Library"
      },
      {
        "name": "QSYS2",
        "attribute": "PROD",
        "description": "System Library for CPI's"
      },
      {
        "name": "QHLPSYS",
        "attribute": "PROD",
        "description": ""
      },
      {
        "name": "QUSRSYS",
        "attribute": "PROD",
        "description": "System Library for Users"
      }
    ],
    "userLibl": [
      {
        "name": "QGPL",
        "attribute": "PROD",
        "description": "General Purpose Library"
      },
      {
        "name": "QTEMP",
        "attribute": "TEST",
        "description": ""
      }
    ],
    "envVariables": {},
    "jobLog": []
  }
}

```

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

### Security Requirements

- [bearerHttpAuthentication](#)

## PUT /api/v1/session

Refresh session settings.

Refresh session settings. The settings affect the remote command and database host server jobs that are tied to the session. Refreshing session settings may result in the ending of existing host server jobs.

### Parameters

#### Authorization (header)

The authorization HTTP header.

**type**  
string

## Request body

The settings for the session.

**Required:** true

### Media types

**application/json**

#### Schema

[“RSEAPI\\_SessionSettings” on page 86](#)

#### Example

```
{
  "resetSettings": true,
  "continueOnError": true,
  "libraryList": [
    "lib1",
    "lib2"
  ],
  "clCommands": [
    "QSYS/CLRLIB LIB(BUILD)"
  ],
  "envVariables": {
    "var1": "var1val",
    "var2": "var2val"
  },
  "sqlDefaultSchema": "lib1",
  "sqlProperties": {
    "auto commit": "true",
    "ignore warnings": "01003,0100C,01567"
  },
  "sqlStatements": [
    "SET PATH = LIB1, LIB2"
  ]
}
```

## Responses

### 204

Successful request, no content.

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)

## POST /api/v1/session

---

Authenticate with user credentials and return an embedded token.

Authenticate with user credentials and return an embedded token to access different RSE APIs. On successful authentication, a token is returned in the Authorization HTTP header. The client must send this token in the Authorization HTTP header when making requests to protected RSE APIs. For example:

```
Authorization: Bearer 4eaa14e6-ea9c-4dde-b6f7-f542b34d5309-60da75d3-3132
```

For optimal performance, specify localhost to access objects located on the same system in which the RSE APIs are hosted on. If accessing objects on a remote system using the RSE APIs, the host servers on the remote system must be enabled for secure communications.

Alternatively, the non-session APIs may be invoked using basic authentication. In this case, the credentials must always be sent on every request.

### Request body

The user credentials to be authenticated.

**Required:** true

### Media types

**application/json**

#### Schema

[“RSEAPI\\_LoginCredentials” on page 77](#)

#### Example

```
{
  "host": "localhost",
  "userid": "user",
  "password": "pwd"
}
```

### Responses

#### 201

Successful request, new resource created.

#### 400

Bad request.

#### 401

Unauthorized request was made.

#### 403

The request is forbidden.

#### 500

Unable to process the request due to an internal server error.

## DELETE /api/v1/session

---

Logout, releasing resources tied to the session.

Logout, releasing resources tied to the session. If a logout is not performed, it will be discarded after a period of idle time (default is 2 hours).

## Parameters

### Authorization (header)

The authorization HTTP header.

**type**

string

## Responses

### 204

Successful request, no content.

### 400

Bad request.

### 401

Unauthorized request was made.

### 403

The request is forbidden.

### 500

Unable to process the request due to an internal server error.

## Security Requirements

- [bearerHttpAuthentication](#)



---

# Components

## Schemas

---

### RSEAPI\_CLCommands

#### Properties

##### **continueOnError**

boolean

##### **description**

Continue processing CL commands if an error is encountered.

##### **default**

false

##### **includeMessageOnSuccess**

boolean

##### **description**

Return CL command messages on success.

##### **default**

false

##### **includeMessageHelpText**

boolean

##### **description**

Return message help text for CL command messages.

##### **default**

false

##### **clCommands**

array

##### **description**

CL command to run.

##### **items**

string

##### **type**

object

##### **description**

List of CL commands to run.

##### **array**

clCommands

## RSEAPI\_DCMCertAppDefTrustRequest

### Properties

#### appDefinitionID

string

#### description

The application definition identifier.

#### certAlias

string

#### description

The certificate label.

### required

- appDefinitionID
- certAlias

### type

object

### description

Add/remove a CA certificate to/from an application definition list of trusted CA certificates.

## RSEAPI\_DCMCertAppDefAssociateRequest

### Properties

#### appDefinitionID

string

#### description

The application definition identifier.

#### certAliases

array

#### description

The certificate label. Maximum of 4.

#### items

string

### required

- appDefinitionID
- certAliases

### type

object

### description

Associate certificate(s) to an application definition.

## RSEAPI\_LoginCredentials

### Properties

#### host

string

#### description

The IBM i server from which objects are to be accessed.

#### default

localhost

#### userid

string

#### description

The user ID.

#### password

string

#### description

The password.

### required

- password
- userid

### type

object

### description

The login credentials.

## RSEAPI\_DCMCertStoreChangePasswordRequest

### Properties

#### certStoreType

string

#### description

The type of the certificate store. Valid values: CMS.

#### certStorePath

string

#### description

Path to certificate store or one of the following special values: \*SYSTEM, \*LOCALCA, \*OBJECTSIGNING, or SIGNATUREVERIFICATION.

#### certStorePassword

string

#### description

The certificate store password. If field omitted or set to null, the system stash will be used.

**certStorePasswordNew**

string

**description**

The new certificate store password.

**daysToExpiration**

integer

**description**

Number of days before password expires.

**default**

0

**required**

- certStorePassword
- certStorePasswordNew
- certStorePath
- certStoreType

**type**

object

**description**

Change certificate store password request.

**RSEAPI\_Settings****Properties****persist**

boolean

**description**

Save settings to property file in persistent storage (hard disk).

**default**

false

**adminUsers**

array

**description**

User IDs that will be designated as an RSE API administrator.

**items**

string

**includeUsers**

array

**description**

User IDs allowed to use RSE API.

**items**

string

**excludeUsers**

array

**description**

User IDs not allowed to use RSE API.

**items**

string

**maxFileSize**

integer (*int64*)

**maximum**

15360000

**minimum**

0

**description**

Maximum size of IFS file data that can be processed (reading or writing) by RSE API.

**default**

3072000

**maxSessions**

integer (*int64*)

**minimum**

-1

**description**

Maximum number of total sessions. The value of -1 indicates there is no limit.

**default**

100

**maxSessionsPerUser**

integer (*int64*)

**description**

Maximum number of sessions allowed on a per-user basis. The value of -1 indicates there is no limit. A value other than -1 must be greater than zero.

**default**

20

**maxSessionInactivity**

integer (*int64*)

**maximum**

7200

**minimum**

30

**description**

Maximum amount of inactive time, in seconds, before an available session is invalidated.

**default**

7200

**maxSessionLifetime**

integer (*int64*)

**description**

Maximum life, in seconds, for a session. The value of -1 indicates there is no limit. A value other than -1 must be greater or equal to 30.

**default**

-1

**maxSessionUseCount**

integer (*int64*)

**description**

Maximum number of times a session can be used before it is invalidated. The value of -1 indicates there is no limit. A value other than -1 must be greater than zero.

**default**

-1

**maxSessionWaitTime**

integer (*int64*)

**minimum**

-1

**description**

Maximum time, in seconds, to wait on a session to become available. The value of -1 indicates there is no limit.

**default**

300

**sessionCleanupInterval**

integer (*int64*)

**maximum**

900

**minimum**

30

**description**

The time interval, in seconds, for how often the session maintenance daemon is run.

**default**

300

**type**

object

**description**

Global settings for RSE API.

**RSEAPI\_FileContent****Properties****content**

string

**type**

object

**description**

The file content.

**array**  
content

## **RSEAPI\_DCMCertRequest**

### **Properties**

#### **certStoreType**

string

#### **description**

The type of the certificate store. Valid values: CMS.

#### **certStorePath**

string

#### **description**

Path to certificate store or one of the following special values: \*SYSTEM, \*LOCALCA, \*OBJECTSIGNING, or \*SIGNATUREVERIFICATION.

#### **certStorePassword**

string

#### **description**

The certificate store password.

#### **certAlias**

string

#### **description**

The certificate label.

### **required**

- certAlias
- certStorePassword
- certStorePath
- certStoreType

### **type**

object

### **description**

Certificate request.

## **RSEAPI\_DCMCertListRequest**

### **Properties**

#### **certStoreType**

string

#### **description**

The type of the certificate store. Valid values: CMS.

#### **certStorePath**

string

**description**

Path to certificate store or one of the following special values: \*SYSTEM, \*LOCALCA, \*OBJECTSIGNING, or SIGNATUREVERIFICATION.

**certStorePassword**

string

**description**

The certificate store password.

**filters**

object

**properties****certAlias**

string

**description**

Alias name filter. A simple generic name can be specified. For example, myCert\*

**certTypes**

array

**description**

Certificate types filter. Valid values: CA, SERVER\_CLIENT.

**items**

string

**daysUntilExpiration**

integer

**description**

Days until expiration filter.

**excludeExpired**

boolean

**description**

Whether to exclude expired certificates.

**description**

One or more combination of filters.

**required**

- certStorePassword
- certStorePath
- certStoreType

**type**

object

**description**

List certificate request.



## RSEAPI\_DCMCertAppDefDisassociateRequest

### Properties

#### appDefinitionID

string

#### description

The application definition identifier.

### type

object

### description

Disassociate certificates from an application definition.

### array

appDefinitionID

## RSEAPI\_SQLRequest

### Properties

#### alwaysReturnSQLStateInformation

boolean

#### description

Always return SQL state information. Default value is whatever has been set for the session.

#### treatWarningsAsErrors

boolean

#### description

Treat SQL warnings as errors. Default value is whatever has been set for the session.

#### sqlStatement

string

#### description

The SQL statement to be run.

### type

object

### description

SQL request to run.

### array

sqlStatement

## RSEAPI\_DCMCertListFilters

### Properties

#### certAlias

string

#### description

Alias name filter. A simple generic name can be specified. For example, myCert\*

**certTypes**

array

**description**

Certificate types filter. Valid values: CA, SERVER\_CLIENT.

**items**

string

**daysUntilExpiration**

integer

**description**

Days until expiration filter.

**excludeExpired**

boolean

**description**

Whether to exclude expired certificates.

**type**

object

**RSEAPI\_DCMCertImportRequest****Properties****certStoreType**

string

**description**

The type of the certificate store. Valid values: CMS.

**certStorePath**

string

**description**

Path to certificate store or one of the following special values: \*SYSTEM, \*LOCALCA, \*OBJECTSIGNING, or SIGNATUREVERIFICATION.

**certStorePassword**

string

**description**

The certificate store password.

**certType**

string

**description**

The certificate type. Possible values: CA, or SERVER\_CLIENT

**certFormat**

string

**description**

The format of the certificate. Possible values: PKCS12, DER, or PEM.

**certAlias**

string

**description**

The certificate label. This property is ignored when certificate type is set to CERTIFICATE\_SIGNING\_REQUEST.

**certData**

string

**description**

Base64-encoded binary data object representing the certificate to be imported.

**certDataPassword**

string

**description**

The password to access the certificate data. This property is only used when the certificate type is set to SERVER\_CLIENT.

**required**

- certData
- certFormat
- certStorePassword
- certStorePath
- certStoreType
- certType

**type**

object

**description**

Import certificate request.

**RSEAPI\_DCMCertExportRequest****Properties****certStoreType**

string

**description**

The type of the certificate store. Valid values: CMS.

**certStorePath**

string

**description**

Path to certificate store or one of the following special values: \*SYSTEM, \*LOCALCA, \*OBJECTSIGNING, or SIGNATUREVERIFICATION.

**certStorePassword**

string

**description**

The certificate store password.

**certFormat**

string

**description**

The format of the certificate. Possible values: PKCS12, DER, or PEM.

**certAlias**

string

**description**

The certificate label.

**certDataPassword**

string

**description**

The password to access the certificate data that is returned. This field is only used when certificate type is SERVER\_CLIENT.

**required**

- certAlias
- certFormat
- certStorePassword
- certStorePath
- certStoreType

**type**

object

**description**

Export certificate request.

**RSEAPI\_SessionSettings****Properties****resetSettings**

boolean

**description**

Replace existing sessions attributes. A value of false will merge settings in request with existing session settings.

**default**

false

**continueOnError**

boolean

**description**

Continue with session processing if an error occurs.

**default**

false

**libraryList**

array

**description**

Library to be added to library list of the remote command host server job tied to session.

**items**

string

**clCommands**

array

**description**

CL command to be run in remote command host server job tied to session.

**items**

string

**envVariables**

object

**description**

Environment variable to set in remote command host server job tied to session.

**additionalProperties**

string

**additionalPropertiesSchema**

string

**sqlDefaultSchema**

string

**description**

Default SQL schema to use when running SQL statements in database host server job.

**sqlTreatWarningsAsErrors**

boolean

**description**

Treat SQL warnings as errors.

**default**

false

**sqlProperties**

object

**description**

Java toolbox JDBC property.

**additionalProperties**

string

**additionalPropertiesSchema**

string

**sqlStatements**

array

**description**

SQL statment to be run in database host server job tied to session.

**items**

string

**type**

object

**description**

The settings for the session.

## Security schemes

---

**bearerHttpAuthentication****Description**

Bearer token authentication.

**Type**

http

**HTTP Authorization scheme**

bearer

**Bearer token format**

Bearer [token]

**basicHttpAuthentication****Description**

Basic authentication.

**Type**

http

**HTTP Authorization scheme**

basic

# Configuration files

This chapter describes the configuration files used by RSE API.

## rseapi.properties

The RSE API uses a properties file, `IBMiRSEAPI.properties`, to store properties that affect RSE API. The property file, if it exists, is stored in `/QIBM/UserData/OS/RSEAPI/rseapi.properties`.

*Table 4. List of RSE API properties*

Property	Default value	Description
<code>com.ibm.rseapi.adminusers</code>		Comma separated list of user IDs that are administrators.
<code>com.ibm.rseapi.excludeusers</code>		Comma separated list of user IDs not allowed to use the RSE API.
<code>com.ibm.rseapi.includeusers</code>		Comma separated list of user IDs allowed to use the RSE API. By default, all users are allowed to use RSE API.
<code>com.ibm.rseapi.maxfilesize</code>	3072000	The maximum size, in bytes, of data that is allowed when reading from files and writing to files. The minimum value is 0 bytes, and the maximum value is 15 megabytes.
<code>com.ibm.rseapi.maxsessioninactivity</code>	7200	A maximum amount of inactive time, in seconds, before a session is reclaimed. The minimum value is 30 seconds, the maximum value is 7200 seconds.
<code>com.ibm.rseapi.maxsessionlifetime</code>	-1	Maximum life for a session. A value of -1 indicates there is no limit. If -1 is not specified, the minimum value is 30 seconds.
<code>com.ibm.rseapi.maxsessions</code>	600	Maximum number of active sessions. A value of -1 indicates there is no limit.
<code>com.ibm.rseapi.maxsessionsperuser</code>	20	Maximum number of active sessions a user can establish. A value of -1 indicates there is no limit. If -1 is not specified, the minimum value is 1.
<code>com.ibm.rseapi.maxsessionusecount</code>	-1	Maximum number of times a session can be used before it expires. A value of -1 indicates there is no limit. If -1 is not specified, the minimum value is 1.
<code>com.ibm.rseapi.maxsessionwaittime</code>	300	Maximum time to wait, in seconds, for a session to become available. A value of -1 indicates to wait indefinitely. A value of 0 indicates that if the session cannot be obtained, to return immediately.

Table 4. List of RSE API properties (continued)

Property	Default value	Description
com.ibm.rseapi.sessioncleanupinterval	300	The time interval, in seconds, for how often the maintenance daemon is run to reclaim idle or expired sessions. The minimum value is 30 seconds, and the maximum value is 900 seconds.

By default, there is no property file. A property file is created when an administrator submits an Administration Services REST request using the POST HTTP method with the `persist` property set to `true`. The following is an example of the payload in the request:

```
{
  "persist": false,
  "adminUsers": ["USER1", "USER2"],
  "includeUsers": ["USER1", "USER2"],
  "excludeUsers": [],
  "maxFileSize": 3072000,
  "maxSessionInactivity": 7200,
  "maxSessionLifetime": 21600,
  "maxSessionUseCount": 1000,
  "maxSessionWaitTime": 300,
  "maxSessions": 100,
  "maxSessionsPerUser": 20,
  "sessionCleanupInterval": 300
}
```

If the `persist` attribute in the request is set to `false`, then the properties will be used for as long as the RSE API server, `admin5`, is running. On a restart of the server, the properties revert to their default values.

If the `persist` attribute in the request is set to `true`, then the properties will be used. In addition, the properties are stored in the file `/QIBM/UserData/OS/RSEAPI/rseapi.properties`. On restart of the server, the properties that have been saved will be used.

A system administrator can create the file manually. Ensure the file is tagged with a CCSID of 1208 (UTF-8). An example of a file is shown below:

```
#RSE API Properties
#Tue Mar 28 18:03:16 CDT 2023
com.ibm.rseapi.sessioncleanupinterval=300
com.ibm.rseapi.maxsessionwaittime=300
com.ibm.rseapi.maxsessioninactivity=7200
com.ibm.rseapi.adminusers=USER1,USER2
com.ibm.rseapi.includeusers=USER1,USER2
com.ibm.rseapi.maxsessionusecount=1000
com.ibm.rseapi.maxsessions=100
com.ibm.rseapi.maxfilesize=3072000
com.ibm.rseapi.maxsessionlifetime=21600
com.ibm.rseapi.maxsessionsperuser=20
```



---

# RSE API Guides

This part of the document goes through step-by-step examples of how to do various things relating to RSE API.



---

# Configure TLS for the admin5 server

## What you'll learn

Learn how you can use the **Configure TLS** wizard to enable TLS for the admin5 server. The wizard is provided by the Web Administration for i interface.

In this example, a self-signed certificate is used. Depending on your needs, there are pros and cons to self-signed versus certificate authority (CA)-signed digital certificates. When you are deciding whether to generate a self-signed certificate or purchase a signed certificate from a CA, consider the following:

- You can easily create self-signed certificates by using the **Configure TLS** wizard. However, these self-signed certificates are not verified by a trusted third party.
- The primary advantage of using certificates from a CA is that the identity of the certificate holder is verified by a trusted third party. The disadvantages include extra cost and administrative effort. If you decide to use a third-party certificate, obtain it from a CA.
- A CA provides a centralized source for posting and obtaining information about certificates, including information about revoked certificates.

## Prerequisites and assumptions

By default, only users with \*ALLOBJ and \*IOSYSCFG special authorities can manage and create web-related servers on the system by using the Web Administration for i interface. A user without the necessary IBM i special authorities to manage or create web-related servers requires an administrator to grant that user permission to a server or group of servers.

## Step 1. Navigate to the admin5 server

Access the Web Administration for i interface from your browser by using the following URL: `http://<host>:2001/HTTPAdmin`, where *<host>* is the host name or an IP address of the IBM i server.

**Note:** If you are unable to connect to the server, check to see whether the HTTP Administration Server is active.

Referring to [Figure 9 on page 94](#):

1. Click the **Manage** tab **(1)**.
2. Click the **Application Servers** sub-tab **(2)**.
3. Click the **Server** selection list **(3)** and select Admin5.



Figure 9. Navigating to the Admin5 server

## Step 2. Start the Configure TLS wizard

In the navigation page as shown in Figure 10 on page 94, click on **Configure TLS** (1). Press **Next** (2).

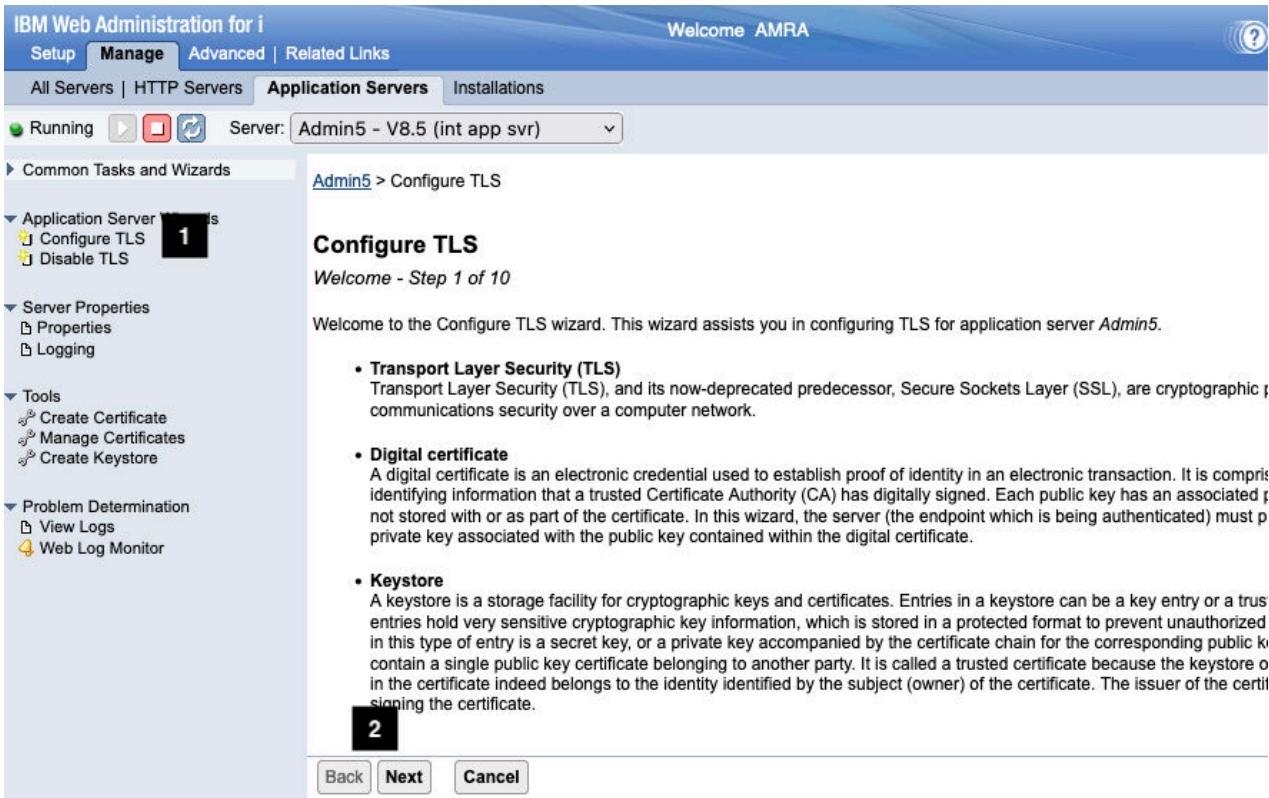


Figure 10. Start Configure TLS wizard

## Step 3. Go through the Configure TLS wizard to configure admin5 server

1. On the page shown in Figure 11 on page 95, you have the ability to specify TLS port and protocol. You can also indicate whether the non-TLS port should be disabled. In this example, the TLS protocol is set to TLSv1.3 (1). Press **Next**.

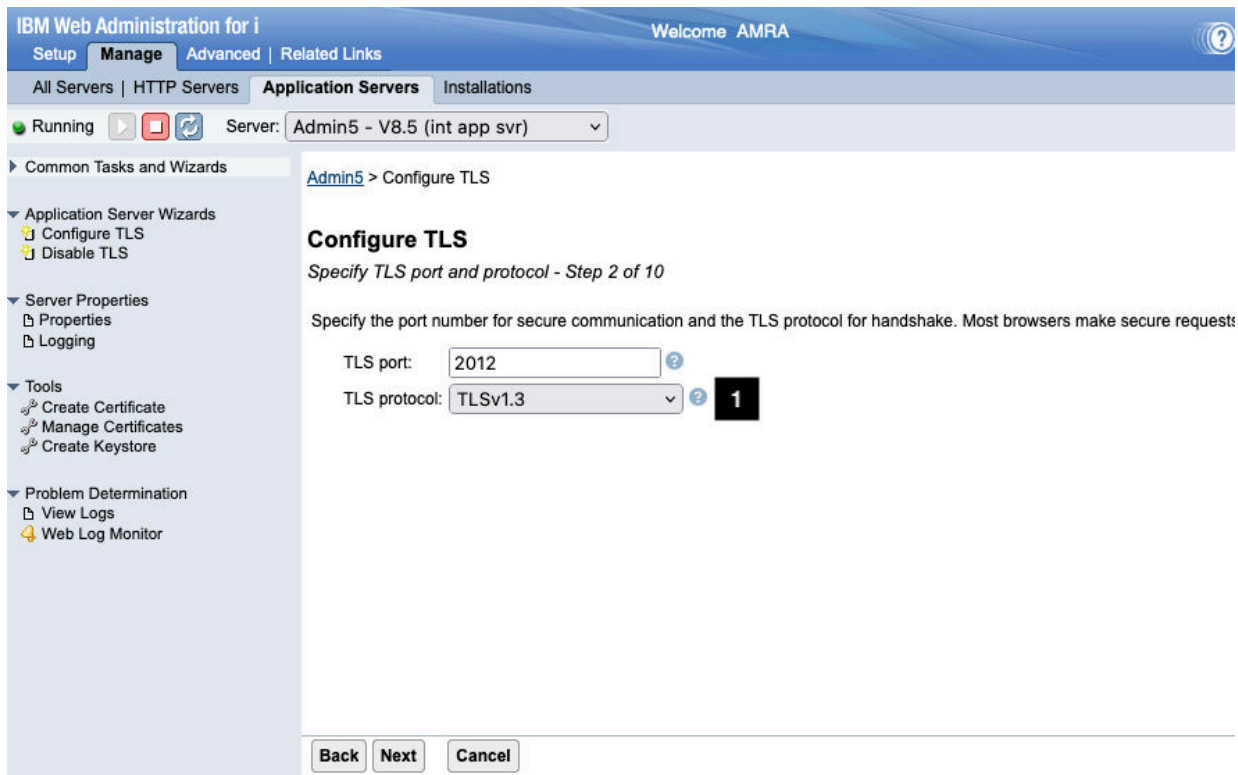


Figure 11. TLS wizard - Specify TLS port and protocol

2. On the panel shown in Figure 12 on page 95, specify the keystore information. In this example, the DCM system store (1) is used. Press **Next**.

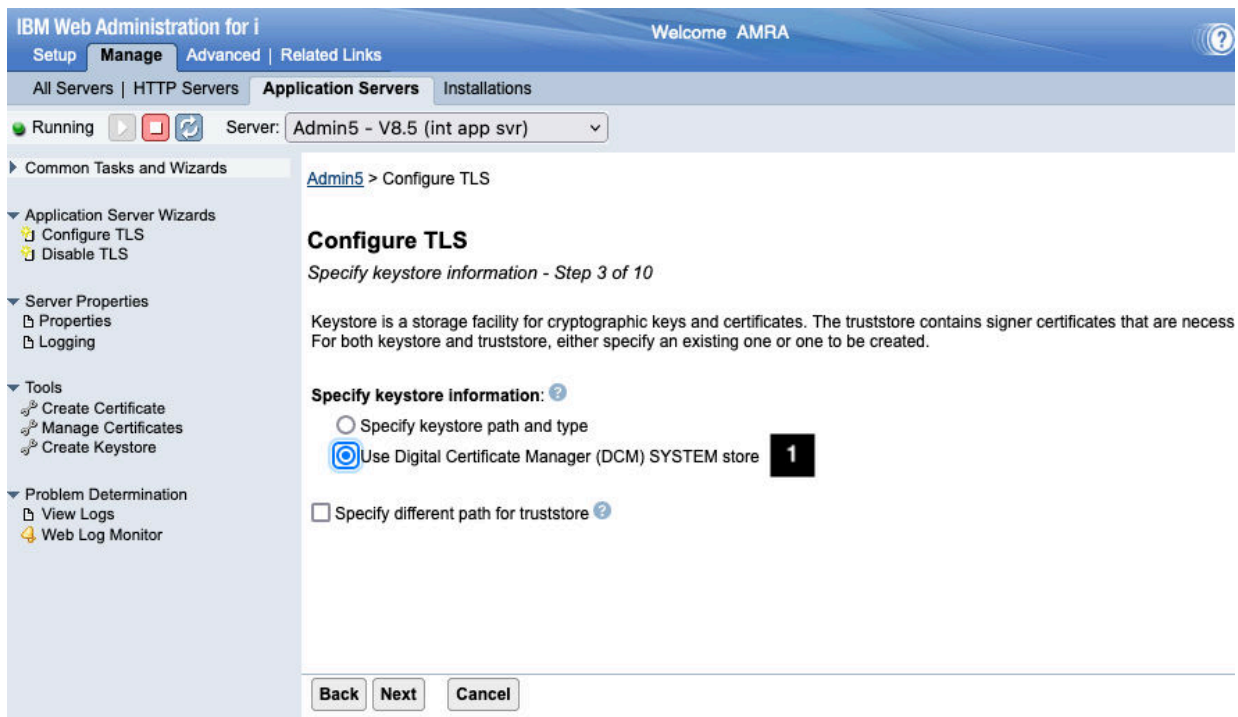


Figure 12. TLS wizard - Specify keystore information

3. On the page shown in Figure 13 on page 96, specify the keystore password (1). Press **Next**.

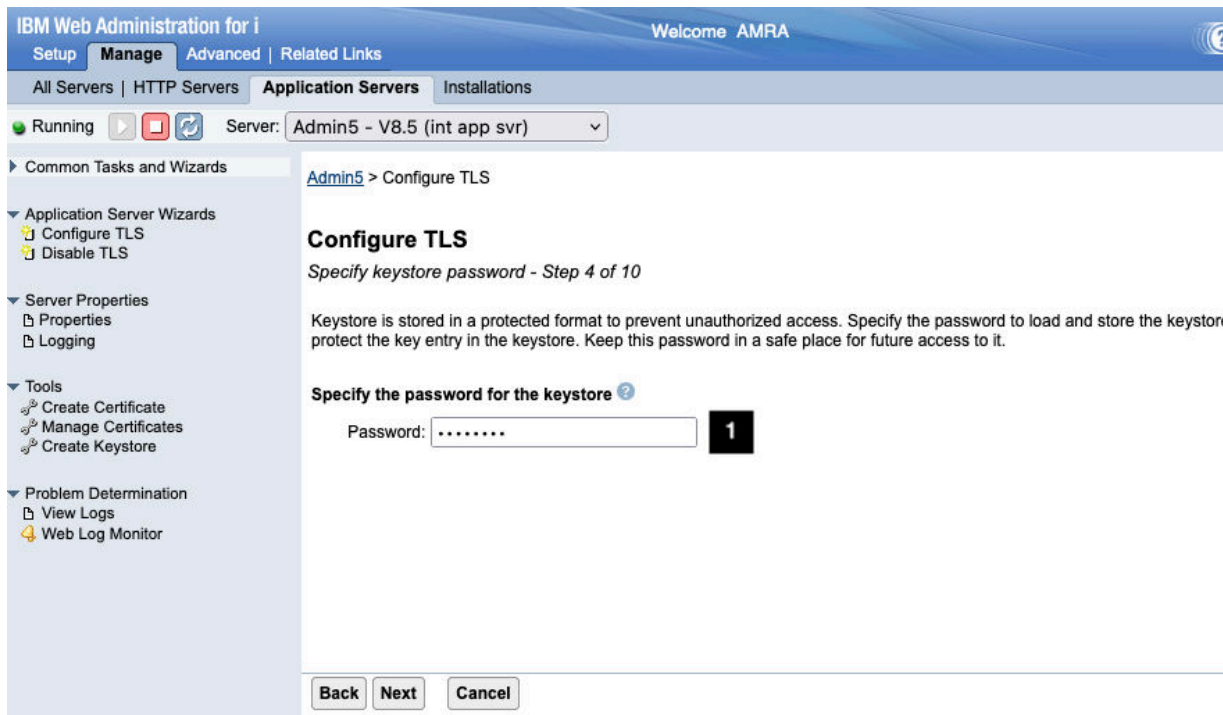


Figure 13. TLS wizard - Specify keystore password

4. On the page shown in Figure 14 on page 96, specify the ciphers for TLS to use. The example uses the default ciphers (1). Press **Next**.

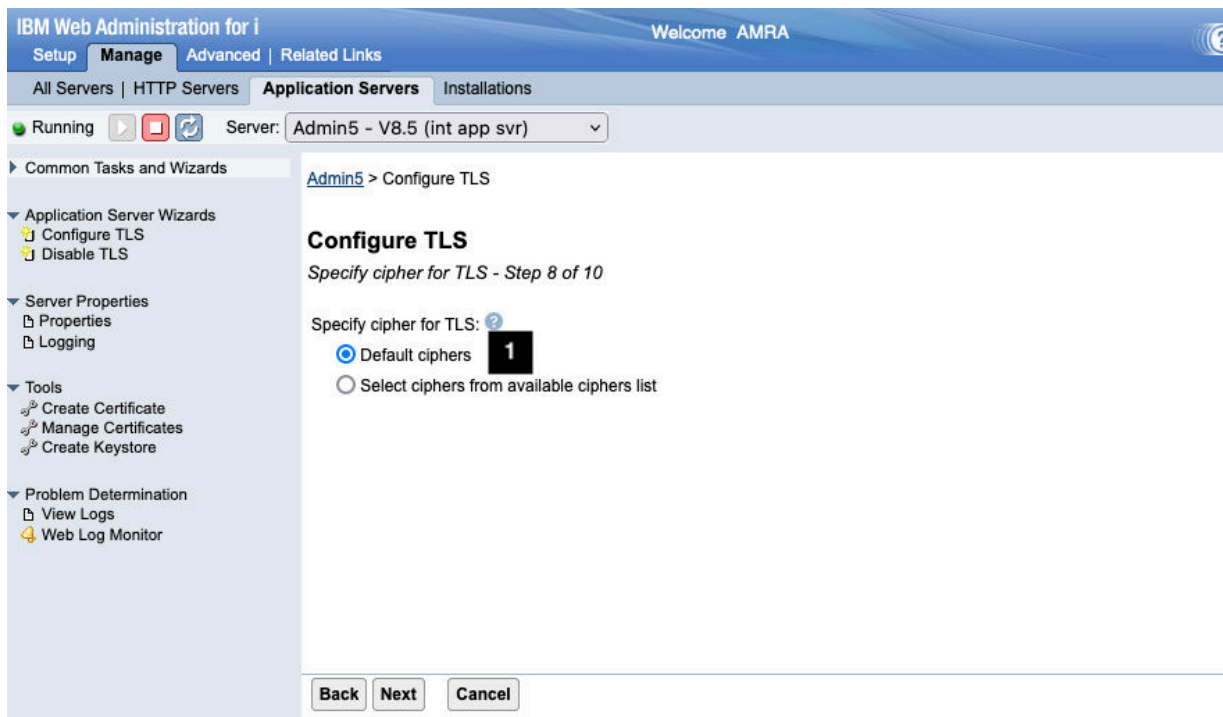


Figure 14. TLS wizard - Specify cipher for TLS

5. On the page shown in Figure 15 on page 97, you have the option to restart server later or to restart server immediately after the wizard completes. The example specifies that the server should be restarted immediately (1) once the wizard completes. Press **Next**.

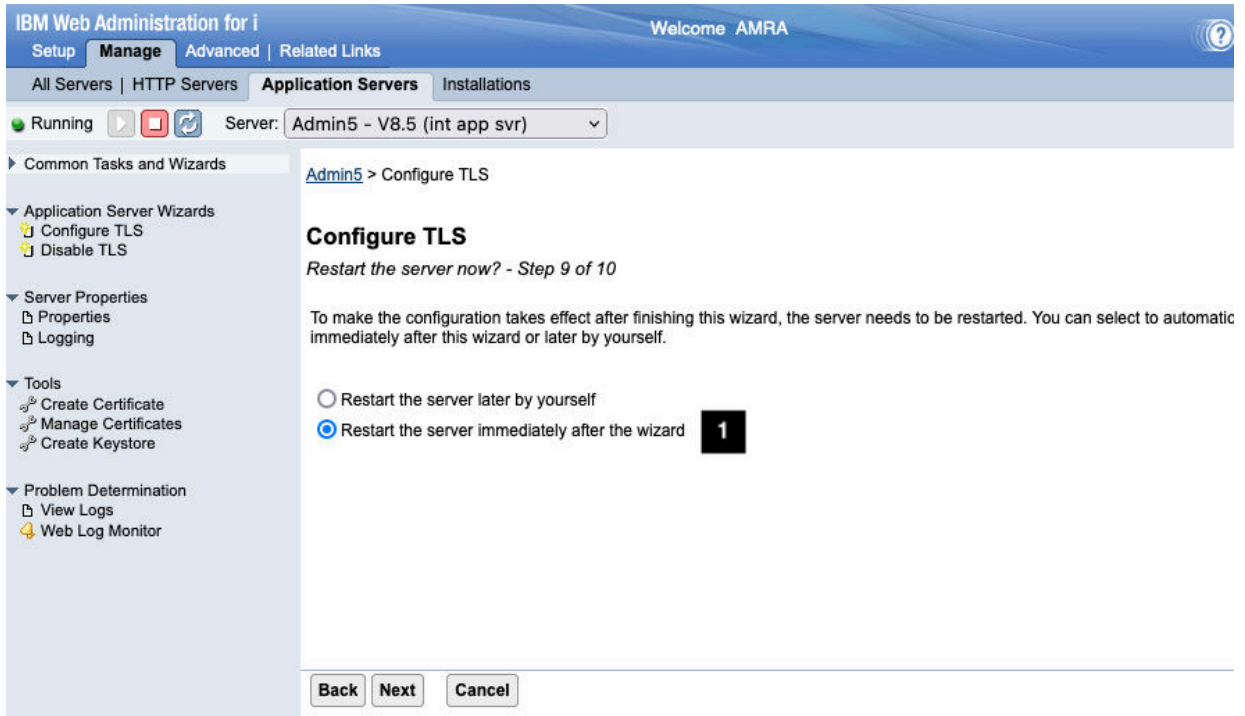


Figure 15. TLS wizard - Restart the server now?

6. A summary page is shown as in Figure 16 on page 97. Note the information shown includes the self-signed certificate name (1). Press **Finish**. After the TLS wizard updates the server, it will restart the server, and the server is now ready to accept RSE API requests.

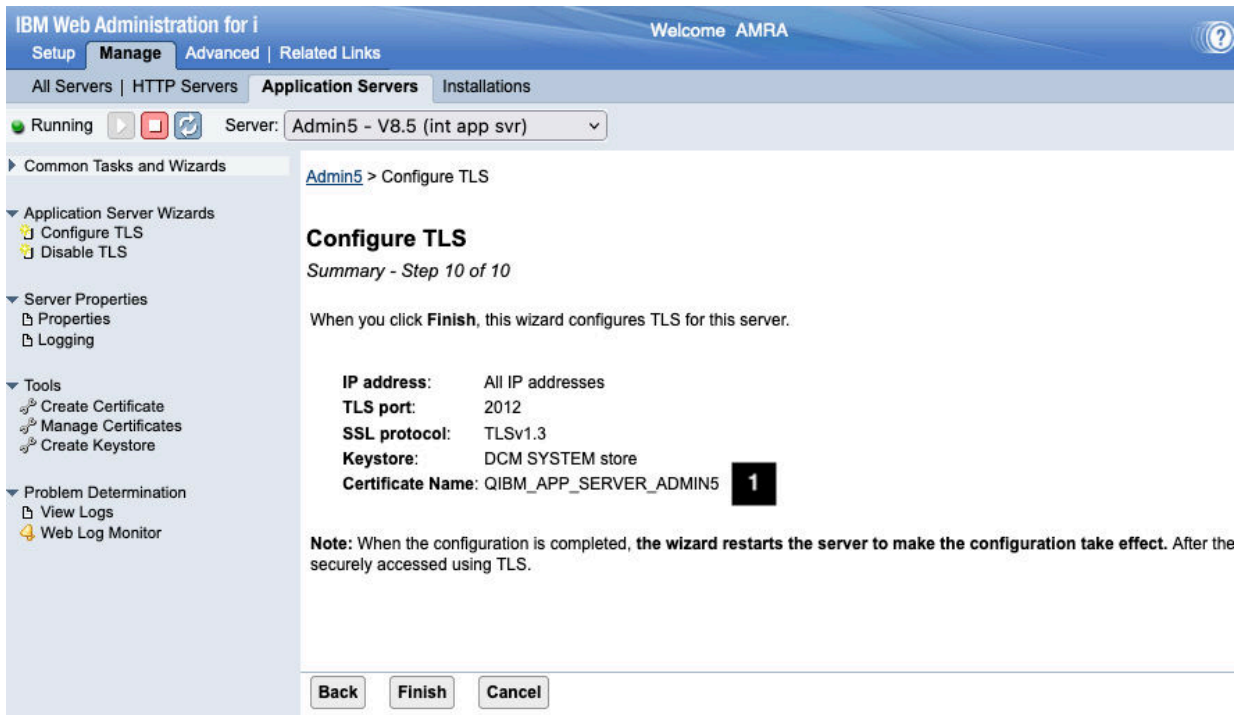


Figure 16. TLS wizard - Summary





# Testing RSE API by using OpenAPI UI

## What you'll learn

The OpenAPI specification, previously known as the Swagger specification, defines a standard interface for documenting and exposing RESTful APIs. RSE API supports a web-based tool, the OpenAPI UI, that enables you to call API operations from within a browser.

## Prerequisites and assumptions

The IBM Web Administration Server must be active. If you have problems connecting to the server, see [“Starting and stopping the RSE API server”](#) on page 9.

Before you can successfully issue API requests, the server must be configured to handle requests over a secure connection. If you need to enable TLS, see [“Configure TLS for the admin5 server”](#) on page 93.

## Step 1. Open the OpenAPI UI page

Using a browser, view the OpenAPI UI page by specifying the following URL: `https://host:2012/openapi/ui/`, where *host* is the host name or IP address of your server. The browser page is shown in Figure 17 on page 99

**Note:** The default TLS port for the admin5 server is 2012. The port can be different on your system depending on whether the system administrator configured a port other than the default.

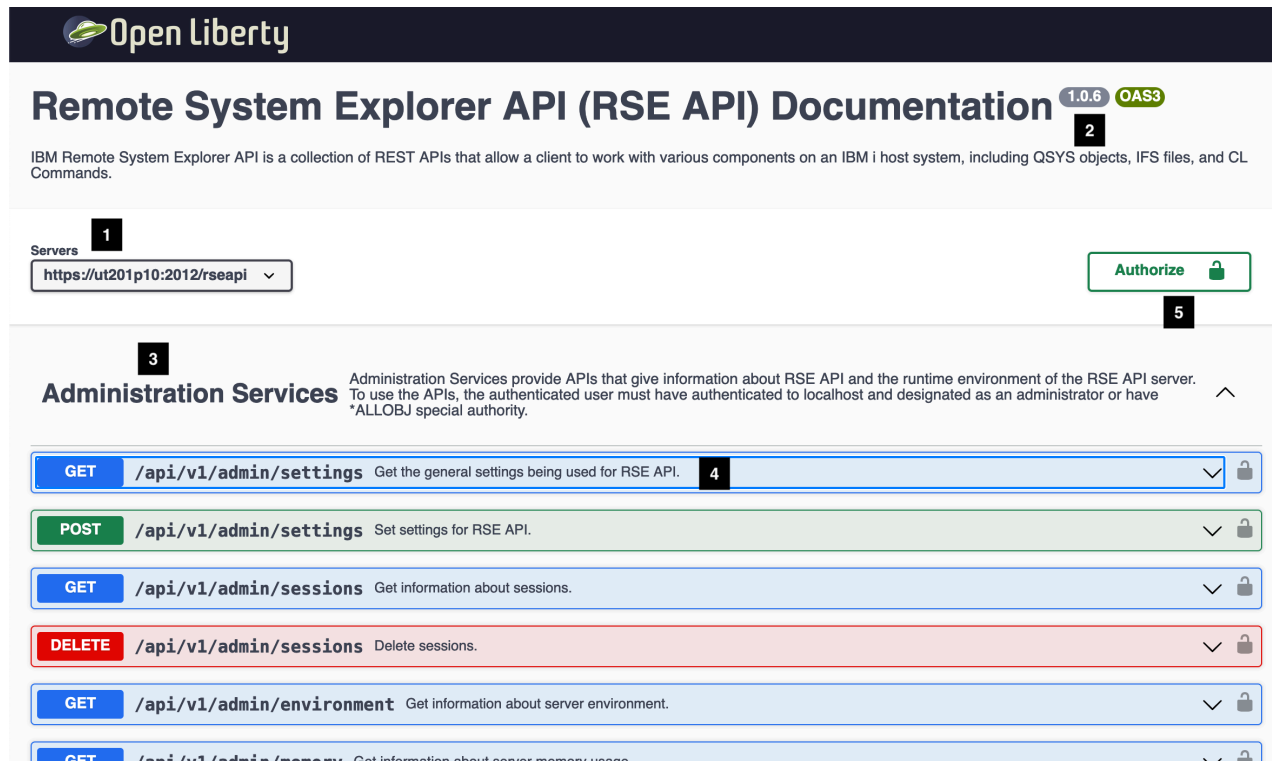


Figure 17. The OpenAPI UI

Some highlights of the OpenAPI UI:

1. The base URL (1) to RSE API.
2. The version (2) of RSE API is shown. In the figure, it is 1.0.6.

3. The RSE API category and description (3). Under each category is a list of APIs.
4. Each row (4) under an RSE API category represents an API. The row shows the HTTP method, the path that would need to be appended to the base URL, and a short description. Clicking the row expands the row to reveal further details about the API and the ability to invoke the API. More on this later.
5. Clicking the security lock (5) results the authentication prompt being displayed.

## Step 2. Authenticate

Authentication in the OpenAPI UI needs to be only done *one* time. Clicking any of the security locks results in the display of an authentication prompt similar to the one shown in Figure 18 on page 100, which shows the available authentication schemes: bearer (1) and basic (2). (The one exception is if you click the security lock corresponding to a session API, in which case only the bearer authentication scheme is displayed.)

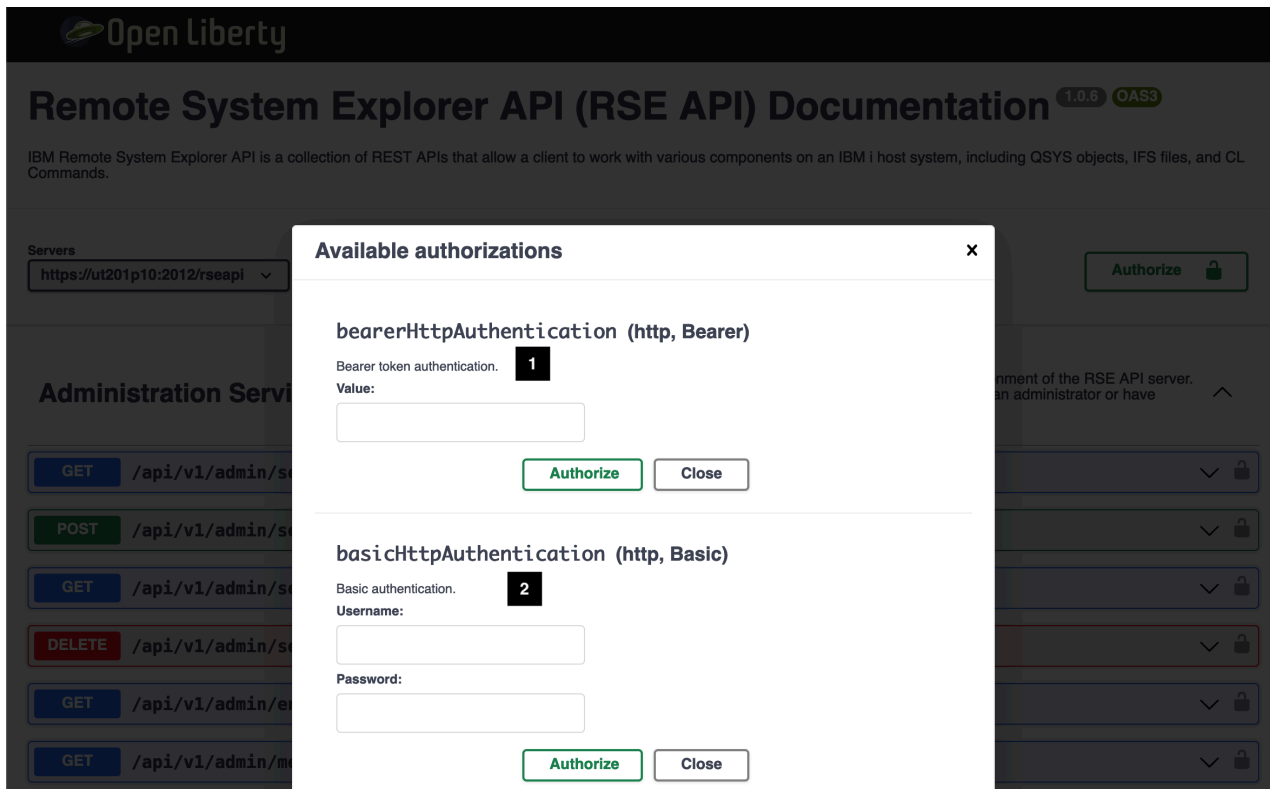


Figure 18. Available authorizations

Click the **Close** button of the **Available authorizations** prompt. The steps to authenticate using HTTP bearer authentication is as follows:

1. To obtain a token, go to the Session Services APIs and select the POST HTTP method as shown in [Figure 19 on page 101](#).

**1** POST /api/v1/session Authenticate with user credentials and return an embedded token.

Authenticate with user credentials and return an embedded token to access different RSE APIs. On successful authentication, a token is returned in the Authorization HTTP header. The client must send this token in the Authorization HTTP header when making requests to protected RSE APIs. For example:

Authorization: Bearer 4aaa14e6-ea9c-4dde-b6f7-f542b34d5309-60da75d3-3132

For optimal performance, specify localhost to access objects located on the same system in which the RSE APIs are hosted on. If accessing objects on a remote system using the RSE APIs, the host servers on the remote system must be enabled for secure communications.

Alternatively, the non-session APIs may be invoked using basic authentication. In this case, the credentials must always be sent on every request.

**2** Parameters

No parameters

**3** Request body required

The user credentials to be authenticated.

Example Value | Schema

```
{
  "host": "localhost",
  "userid": "user",
  "password": "pwd"
}
```

**4** Responses

Code	Description	Links
201	Successful request, new resource created.	No links
400		No links

**5** Try it out

Figure 19. Session API: POST HTTP method

The format of the page to test APIs is the same for all the APIs. The following describes the POST HTTP method for the session API:

- A description **(1)** of the API is shown.
- Parameters **(2)** are listed if there are any. The value of the parameters can be obtained from the URL path, the query string, or an HTTP header.
- The request body **(3)** is the payload that is sent to the server. Not all APIs have payloads. In the session API, an example is shown of the format of payload.
- Possible HTTP status codes **(4)** that can be returned by the server are shown.
- In order to test the API, you need to click the **Try it out (5)** button.

Click the **Try it out (5)** button.

- The page becomes input capable as shown in [Figure 20 on page 102](#). You can now specify your credentials **(1)**.

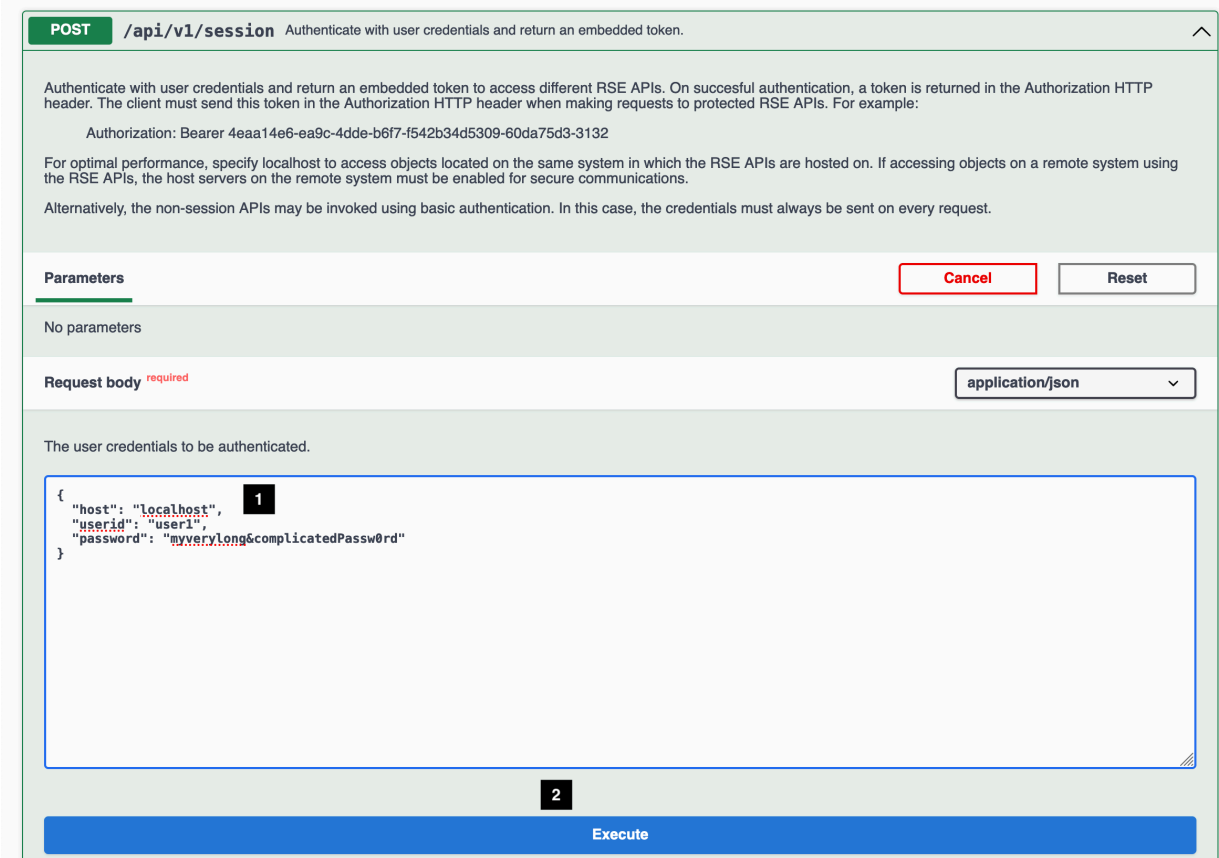


Figure 20. Session API: POST HTTP method - Try it out

Click the **Execute** (2) button to send the request to the server.

3. The page will refresh and include the request and response as shown in Figure 21 on page 102.

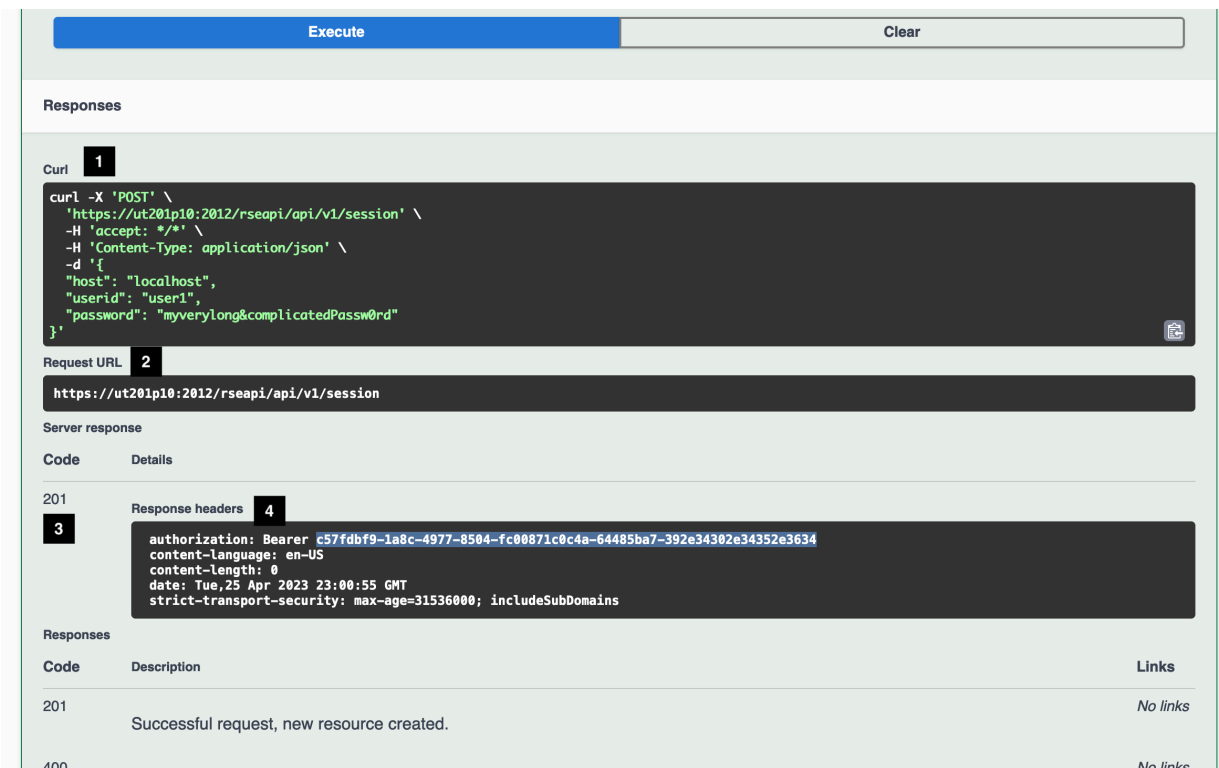
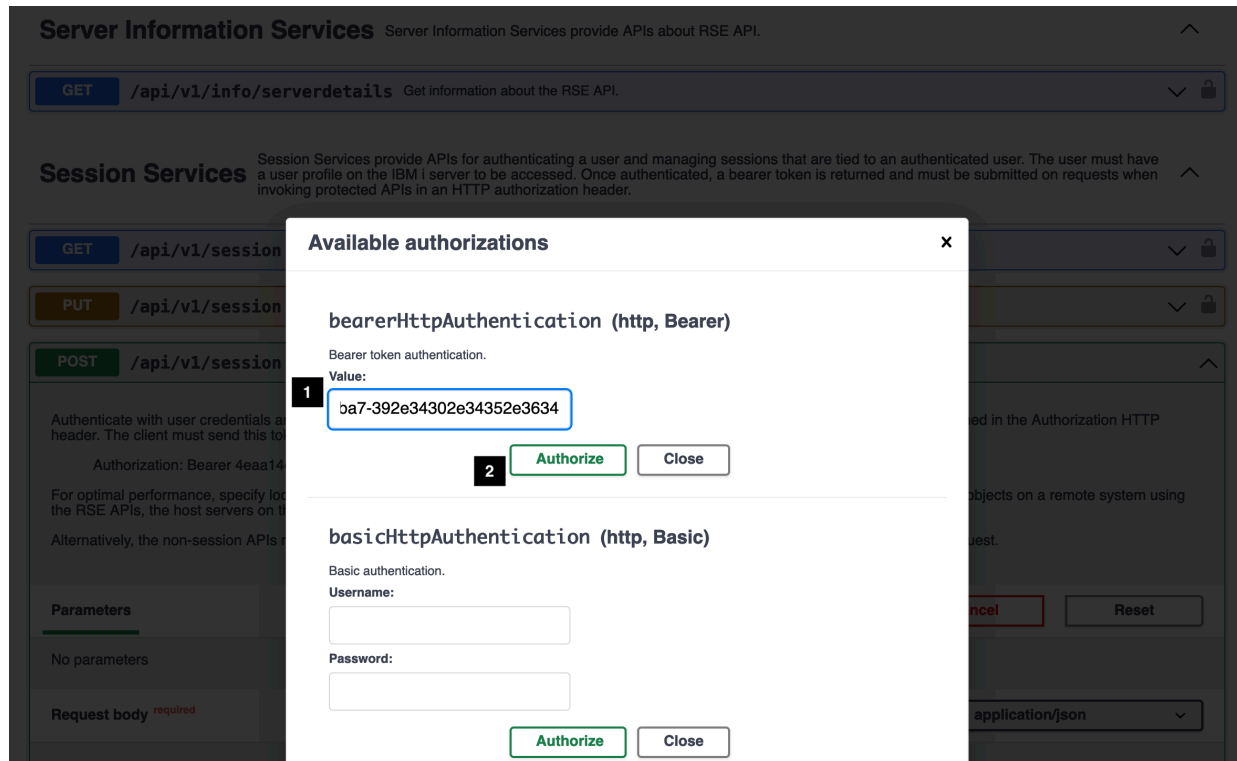


Figure 21. Session API: POST HTTP method - Request and response

- The request in cURL format (1) is shown. cURL stands for client URL, and is a command line tool that is used to transfer data to and from a server. A user can copy and paste to command line to duplicate the request, assuming cURL is installed on their workstation.
- The request URL is shown (2), which is constructed from the base path and the API path.
- The server returns a response. The HTTP status code (3) is shown, as well as HTTP headers in the response (4). The token is returned in the Authorization HTTP header of the response. There is no payload in the response.

Copy the token and click one of the security locks to show the **Available authorizations** prompt in which you can paste the authentication token in the input field.

4. Paste the authentication token in the input field (1) as shown in [Figure 22 on page 103](#). Then click (2) **Authorize** to save the token for the session.



*Figure 22. Setting authentication token - before clicking Authorize*

5. The session token is saved and you will see **Logout** (1) button in addition to the **Close** (2) button as shown in [Figure 23 on page 104](#). Clicking the **Logout** button will enable you to set a new token.

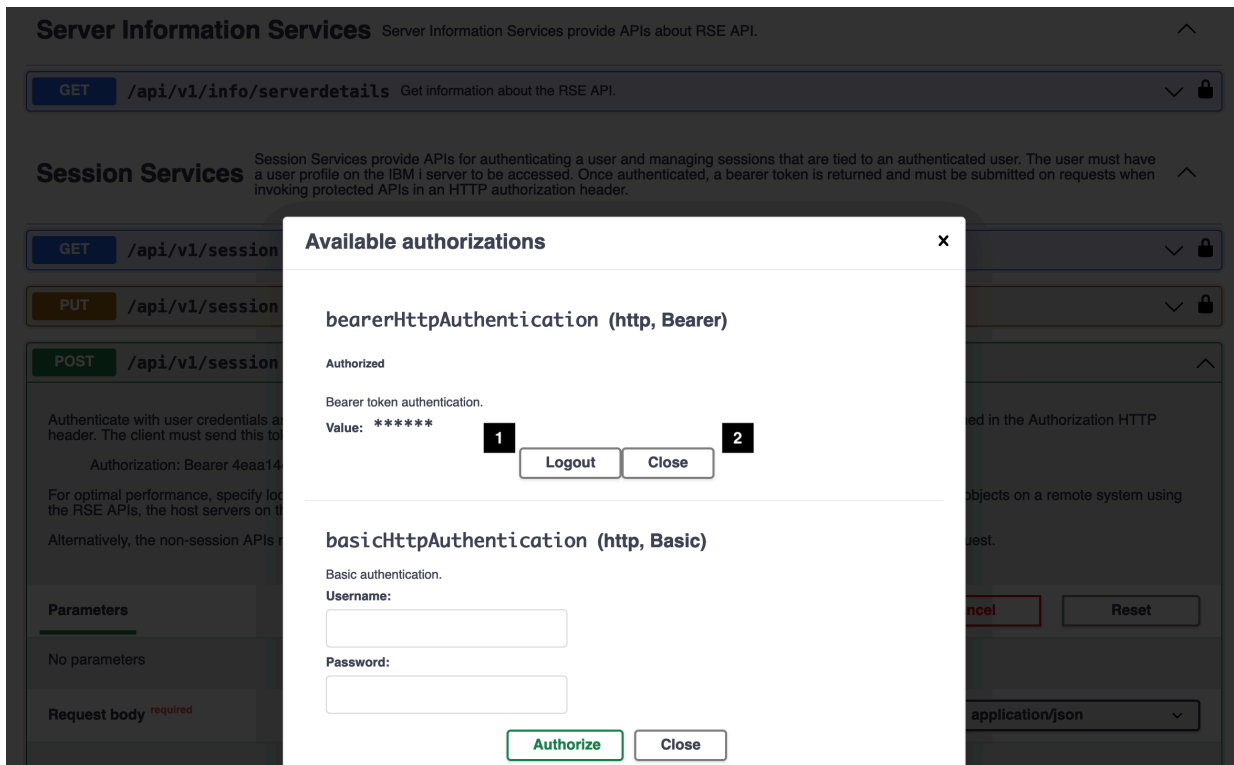


Figure 23. Setting authentication token- after clicking Authorize

Click **Close** button to exit the authorization prompt. You are now able to use any of the APIs as long as the token is valid.

Similarly, authenticating by using basic authentication is simply a matter of clicking any of the security locks to show the **Available authorizations** prompt, specifying your user name and password, and clicking **Authorize** to save the credentials. Click **Close** button to exit the authorization prompt.

### Step 3. Use CL Command Services to create a library with a source physical file

In this step, the PUT HTTP method of the CL Command Services API is used create a library. It is assumed that you have authenticated. Navigate to the CL Command Services APIs, click the PUT HTTP method, then click the **Try it out** button. You should see a web page similar to [Figure 24 on page 105](#).

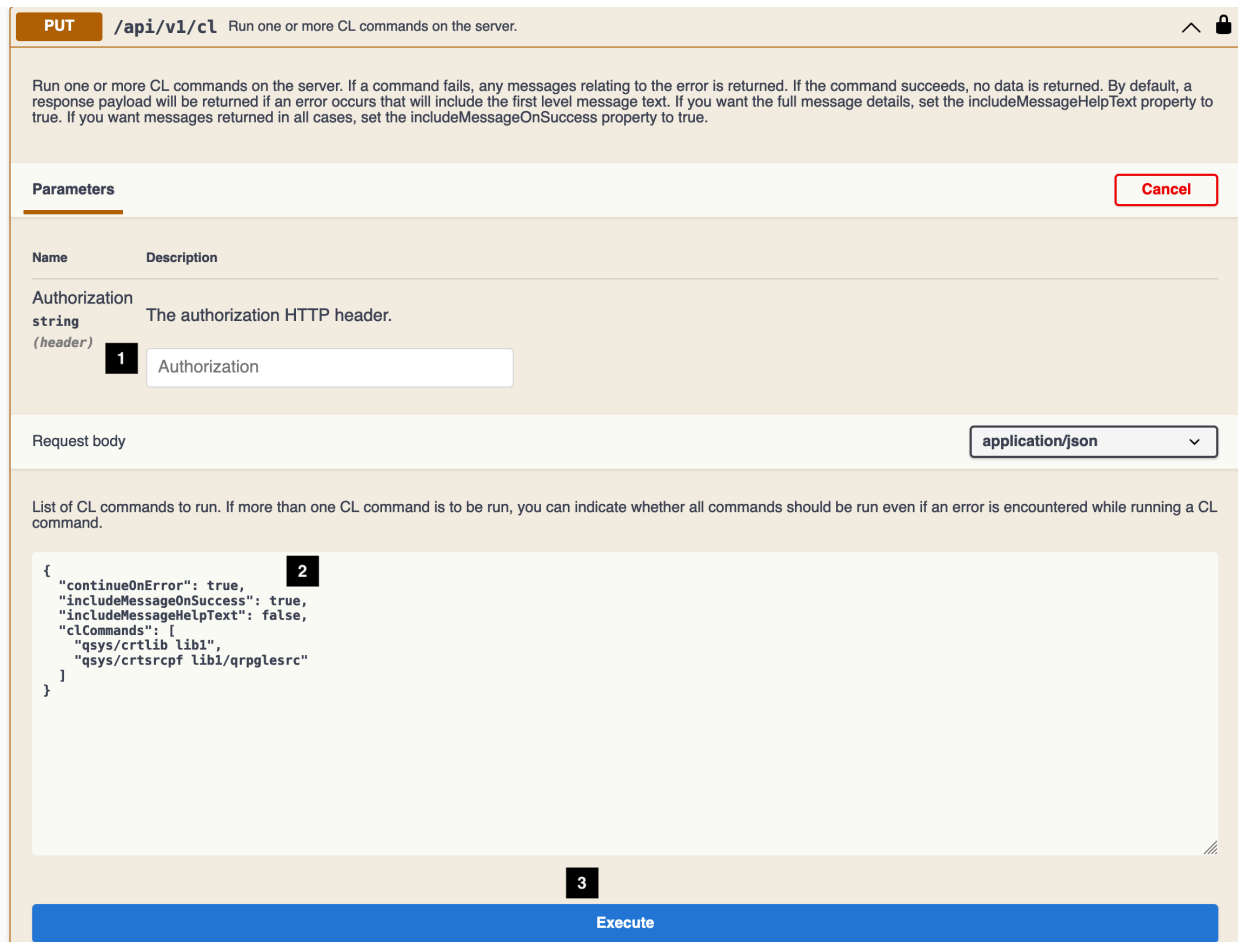


Figure 24. CL Command Services - PUT HTTP method

- Since authentication information has already been set (by clicking on the security lock and setting authentication information), you can ignore the **Authorization** HTTP header field (1) on the API panels.
- An example of the request body (2) is shown. You can add or remove CL commands.

Click the **Execute** (3) button to send the request to the server. The next panel will show the request and response as shown in [Figure 25 on page 106](#)

Responses

Curl

```
curl -X 'PUT' \
  'https://ut201p10:2012/rseapi/api/v1/cl' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer c57fdbf9-1a8c-4977-8504-fc00871c0c4a-64485ba7-392e34302e34352e3634' \
  -H 'Content-Type: application/json' \
  -d '{
    "continueOnError": true,
    "includeMessageOnSuccess": true,
    "includeMessageHelpText": false,
    "clCommands": [
      "qsys/crtlib lib1",
      "qsys/crtsrcpf lib1/qrpglesrc"
    ]
  }'
```

Request URL

https://ut201p10:2012/rseapi/api/v1/cl

Server response

Code	Details
200	<p>Response body <b>2</b></p> <pre>{   "totalIssued": 2,   "totalSuccesses": 2,   "totalFailures": 0,   "commandOutputList": [     {       "success": true,       "command": "qsys/crtlib lib1",       "output": [         "CPC2102: Library LIB1 created. "       ]     },     {       "success": true,       "command": "qsys/crtsrcpf lib1/qrpglesrc",       "output": [         "CPC7301: File QRPGLSRC created in library LIB1. "       ]     }   ] }</pre> <p>Response headers</p> <pre>content-language: en-US content-length: 298 content-type: application/json;charset=utf-8</pre>

Figure 25. CL Command Services - PUT HTTP method response

1. The HTTP status code in response is 200 **(1)**.
2. The response body **(2)** is shown. Because the request attribute `includeMessageOnSuccess` is set to `true`, a response body is returned. If it was set to `false`, you will get an HTTP status code of 201 and a response body is not returned, assuming that all the CL commands ran successfully.

#### Step 4. Use IFS Services to list objects in a library

In this step, the IFS list API is used to list all objects that start with the letter Q in a library. It is assumed that you have authenticated. Navigate to the IFS Services APIs, click the GET HTTP method with the path `/api/v1/ifs/list`, then click the **Try it out** button. You should see a web page similar to [Figure 26](#) on page 107.



**GET** /api/v1/ifs/list Gets a list of objects in the specified path. ^

Gets a list of objects in the specified path. The information returned includes the name, whether object is a directory, the description, and the object subtype. For objects that are not in the QSYS.LIB file system, any part of the path may contain an asterisk (\*), which is a wildcard that means zero or more instances of any character. For example, a path of /tmp/am\*1.txt will return all objects in directory /tmp that have names that begin with 'am' and end with '1.txt'. For QSYS.LIB objects, only generic names may be specified in any part of the path. A generic name is a character string that contains one or more characters followed by an asterisk. For example, /qsys.lib/am\*.lib will return all libraries that have names that start with 'am'. Another example is /qsys.lib/am\*.\*, which would return all objects that start with 'am'. This would be equivalent to specifying a path of /qsys.lib/am\*.

**Parameters** Cancel

Name	Description
Authorization string <small>(header)</small>	The authorization HTTP header. <input type="text" value="Authorization"/>
path * required string <small>(query)</small>	The working directory. For example, /u/IBM/test <input type="text" value="/qsys.lib/amra.lib/q*"/> 1
subtype string <small>(query)</small>	Subtype of objects to return. Valid values include a specific object type (*LIB, *FILE, *PGM, *OUTQ, etc.) or *ALL. Note that many file system objects do not have a subtype. For example, any Root, QOpenSys or UDFS object. <input type="text" value="subtype"/> 2
includehidden boolean <small>(query)</small>	Whether to show hidden files. <input type="text" value="false"/> 3

**Execute**

Figure 26. IFS Services - List objects

- Specify a path (1) you want to search.
- You can specify a subtype (2). In this example a subtype is not set.

Click the **Execute** (3) button to send the request to the server. The next panel will show the request and response as shown in [Figure 27 on page 108](#)

Responses

Curl

```
curl -X 'GET' \
  'https://ut201p10:2012/rseapi/api/v1/ifs/list?path=%2Fqsys.lib%2Ffamra.lib%2Fq%2A&includehidden=false' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer c57fdbf9-1a8c-4977-8504-fc00871c0c4a-64485ba7-392e34302e34352e3634'
```

Request URL **1**

```
https://ut201p10:2012/rseapi/api/v1/ifs/list?path=%2Fqsys.lib%2Ffamra.lib%2Fq%2A&includehidden=false
```

Server response

Code Details

200 **2**

Response body **3**

```
{
  "objects": [
    {
      "path": "/QSYS.LIB/AMRA.LIB/QCLSRC.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/AMRA.LIB/QCSRC.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/AMRA.LIB/QRPGLESRC.FILE",
      "description": "",
      "isDir": true,
      "subType": "PF-SRC"
    },
    {
      "path": "/QSYS.LIB/AMRA.LIB/QSQDSRC.FILE",
      "description": "SQL PROCEDURES",
      "isDir": true,
      "subType": "PF-SRC"
    }
  ]
}
```

Response headers

```
content-language: en-US
content-length: 505
content-type: application/json;charset=utf-8
date: Tue, 25 Apr 2023 23:58:15 GMT
strict-transport-security: max-age=31536000; includeSubDomains
```

Figure 27. IFS Services - list response

1. The path you specify is a query string parameter (**1**), and thus the path is specified as part of the URL for the request.
2. The HTTP status code in response is 200 (**2**).
3. The response body (**3**) is shown. The list of objects that match the search criteria is returned.

### Step 5. Use IFS Services to get detailed information about an object

In this step, the IFS Services API that uses the GET HTTP method with a path of `/api/v1/ifs/{path}/info` is used to return information about an object. It is assumed that you have authenticated. Navigate to the IFS Services APIs, click the GET HTTP method with the path `/api/v1/ifs/{path}/info`, followed by clicking on the **Try it out** button. You should see a web page similar to [Figure 28 on page 109](#).

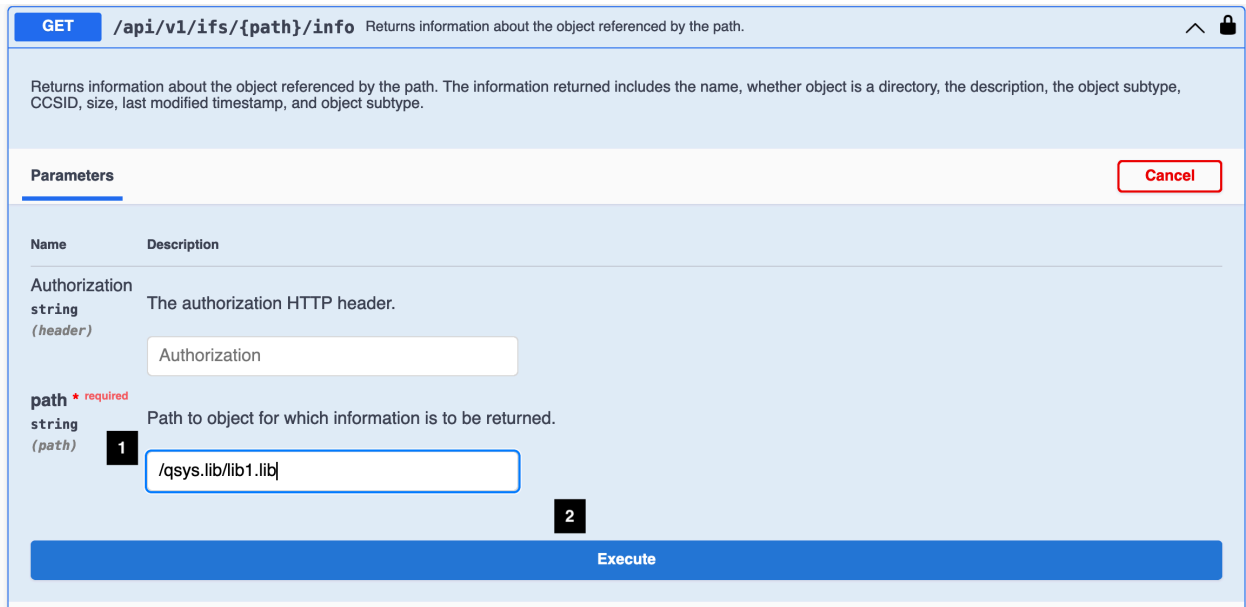


Figure 28. IFS Services - get object information

- Specify a path (1) of the object.

Click the **Execute** (2) button to send the request to the server. The next page will show the request and response as shown in [Figure 29 on page 109](#)

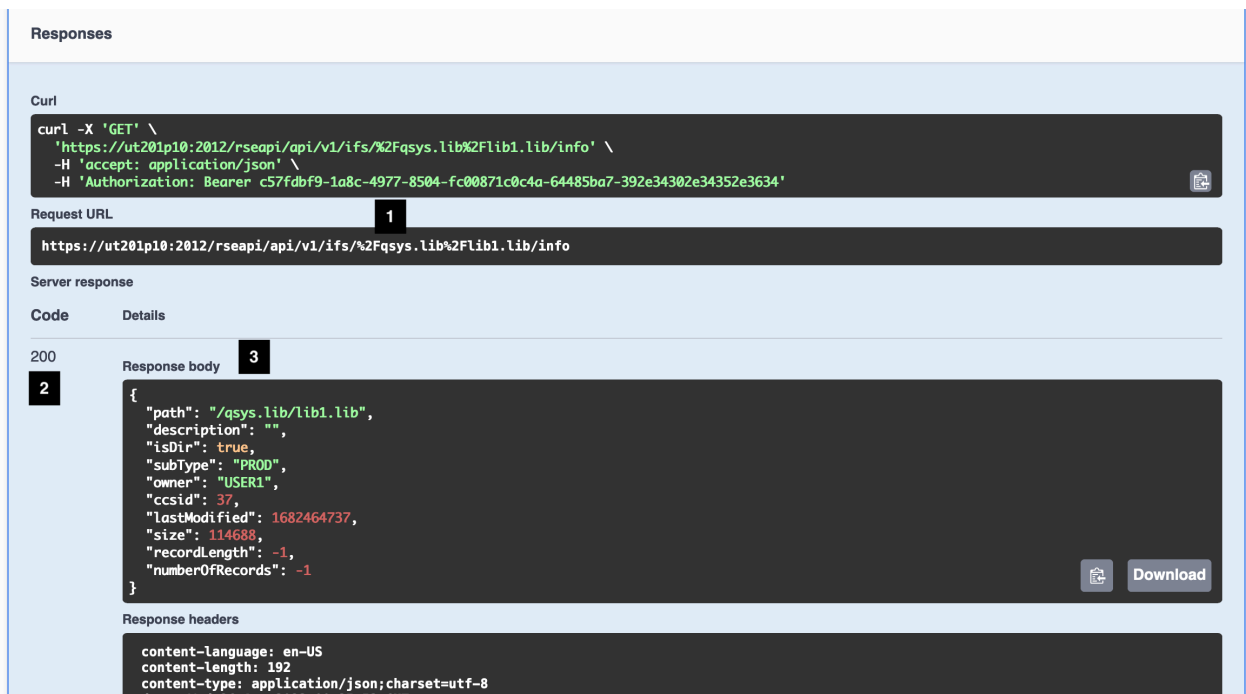


Figure 29. IFS Services - get object information response

1. The path that you specify is a path parameter (1), and thus the path of the object is a component of the URL path.
2. The HTTP status code in response is 200 (2).
3. The response body (3) is shown. The information for the specified object is returned. A value of -1 means that the attribute is not applicable to the object.



## Notices

---

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation  
Software Interoperability Coordinator, Department YBWA  
2800 37th Street NW  
Rochester, MN 55901-4441  
U.S.A.

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

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

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

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

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

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

#### COPYRIGHT LICENSE:

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

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

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

---

IBM, the IBM logo, and [ibm.com](http://ibm.com)<sup>®</sup> are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

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

Java and all Java-based trademarks and logos are trademarks of Oracle, Inc. in the United States, other countries, or both.

Linux<sup>®</sup> is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.





# Glossary

---

**admin server**

The HTTP server and application server instances shipped on IBM i. The application servers are used to host IBM supplied web applications. Currently, there are 5 web application servers, named admin1, admin2, admin3, admin4, and admin5.

**ANSI**

American National Standard for Information Systems

**API**

Application Programming Interface

**attachment**

Data that is attached to a message on the wire, separately from the SOAP envelope. Attachments are often used for sending large files or images.

**authentication**

Confirming the identity of a user. The most common form of authentication is user ID and password, such as through basic authentication. When a user is authenticated, the source of a request is represented as a Subject object at run time.

**authorization**

Determining whether a user has access to a specific role within the system.

**CA - certificate authority**

A Certificate Authority (CA) is a trusted central administrative entity that can issue digital certificates to users and servers. The trust in the CA is the foundation of trust in the certificate as a valid credential.

**certificate**

A credential used as an identity of proof between the server and client. It consists of a public key and some identifying information that a certificate authority (CA), an entity to sign certificates, has digitally signed. Each public key has an associated private key and the server must prove that it has access to the private key associated with the public key contained within the digital certificate. A self-signed certificate means it is signed by the server itself. If a self-signed certificate is specified to a server, clients might not trust the connection. To obtain a signed certificate from a public CA, you need to generate a Certificate Signing Request (CSR) and send it to the CA. After a certificate is returned, it is imported to your keystore.

**DoS**

A denial of service (DoS) attack is a malicious attempt to flood a server with traffic, rendering the web site or resource inaccessible.

**HTTP**

Hypertext Transfer Protocol (HTTP) is a request/response-based protocol that permits clients to interact with servers.

**IBM Toolbox for Java**

See JT400.

**IoT**

The Internet of Things (IoT) is the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data.

**Java Toolbox**

See JT400.

**JT400 - IBM Toolbox for Java**

A set of Java classes that allows Java programs to access data on your IBM i. It's based on open-source package JTOpen.

## **JSON**

JavaScript Object Notation. Lightweight data-interchange format that is built on a collection of name/value pairs alongside ordered lists of values.

## **keystore**

A storage facility for cryptographic keys and certificates. A private key entry in a keystore file holds a cryptographic private key and a certificate chain for the corresponding public key. A private key entry can be specified to a server when configuring SSL. A trusted certificate entry contains a public key for a trusted party, normally a CA. A trusted certificate is used to authenticate the signer of certificates provided by a server or client. The keystore types that the Web Administrator for i GUI supports are: JKS, JCEKS, PKCS12, and CMS. Additionally, the Digital Certificate Manager (DCM) \*SYSTEM is also supported.

## **OpenAPI**

The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. OpenAPI started as Swagger (see Swagger) before it was made into a standard specification and branded as OpenAPI.

## **REST - Representational State Transfer**

A style of software architecture for distributed hypermedia systems such as the World Wide Web. REST strictly refers to a collection of architectural principles. The term is also often used in a loose sense to describe any simple interface that transmits domain-specific data over HTTP without an additional messaging layer such as SOAP or session tracking via HTTP cookies.

## **secure endpoint URL**

Endpoint beginning with https

## **SSL**

Secure Sockets Layer. See TLS.

## **Swagger**

An open specification for defining REST APIs. A Swagger document is the REST API equivalent of a WSDL document for a SOAP-based web service. The Swagger document specifies the list of resources that are available in the REST API and the operations that can be called on those resources.

## **TCPIP**

Transmission Control Protocol/Internet Protocol

## **TLS**

Transport Layer Security (TLS) protocol provides data encryption, data origin authentication, and message integrity. TLS is based on the Secure Sockets Layer (SSL) protocol and is defined by the Internet Engineering Task Force (IETF) in RFCs such as 2246 (TLSv1.0), 4346 (TLSv1.1), and 5246 (TLSv1.2) and 8446 (TLSv1.3). SSL was originally defined as a proprietary protocol, not by the IETF. Since TLS evolved from SSL, the two terms are usually used interchangeably.

## **UI**

A user interface is the part of a software product that your customer actually sees. A user interface may include the layout of display screens or printed output, displayed or printed text, commands, online help, and messages.

## **WAR - Web Application Archive**

A file used to distribute a collection of Java Servlets, Java classes, XML files, static web pages, and other resources that together constitute a web application.

## **Web Administration GUI**

Provides a web-based server management and creation interface for IBM i. Web Administration GUI is rich in function, examples, error-checking, and provides many easy-to-use wizards to help users accomplish many difficult tasks. The GUI supports many different web technologies and helps the user integrate these technologies into a useful production ready web environment.

## **web service**

A self-contained software component with a well-defined interface that describes a set of operations that are accessible over the Internet. Web services are either SOAP-based or REST-based web services.

**wire**

All the underlying components that are responsible for physically sending or receiving a message on the web.

**XML**

eXtensible Mark-up Language



---

# Index

## A

Authentication  
  Authorization header [12, 13](#)  
  Basic [13](#)  
  Bearer [12](#)  
Authorization [13](#)

## C

configuration files  
  rseapi.properties [89](#)

## H

HTTP  
  group PTF [19](#)  
HTTP Administration Server  
  Starting and stopping [9](#)

## I

installation  
  package [7](#)  
  prerequisites [7](#)

## J

JVM  
  dumps [24](#)

## L

logging  
  HTTP access logging [22](#)  
  Server tracing [21](#)

## P

performance tuning  
  load balancing [18](#)  
  network [17](#)  
  RSE API [17](#)  
Properties  
  Security [14](#)  
PTF [19](#)

## R

Role  
  Administrator [14](#)  
RSE API  
  categories [4](#)  
  concepts [3](#)  
  getting started [5](#)

RSE API (*continued*)  
  overview [3](#)

## S

Serviceability  
  Updates [19](#)

## T

TLS  
  certificate recommendations [11](#)  
  cipher recommendations [12](#)  
  private key recommendations [11](#)  
  protocol recommendations [12](#)





