F

REST interface

An interface is provided for Web Gateway that allows you to administer an appliance without being logged on to the standard user interface. This alternative interface is known as the *REST (Representational State Transfer) interface*.

Using the REST interface, you can perform various kinds of activities on a particular appliance or on others that are connected to it.

- **Actions** — Turn off an appliance, restart it, flush the cache, create a configuration backup, and perform several other activities
- **File handling** — Access system, log, and troubleshooting files to perform activities such as downloading, modifying, or deleting
- **Policy configuration** — Configure settings for engines and rule actions, manage rule sets and lists by performing activities such as enabling, adding, deleting, exporting, or importing
- **Updates** — Perform manual engine updates and trigger automatic yum and engine updates

Running a suitable script is the usual way to perform these activities.

Contents

Prepare use of the REST interface

To let users work with the REST interface, you need to enable it on the standard user interface of an appliance and permit access to it.

Tasks

- *Enable use of the interface* on page 880
  You can enable the use of the REST interface for completing administration activities on an appliance.
- *Give permission to access the interface* on page 880
  You must add permission to access the REST interface to an administrator role for those users who are supposed to work with the interface.

Enable use of the interface

You can enable the use of the REST interface for completing administration activities on an appliance.

Task

1. Select **Configuration** | **Appliances**.
2. On the appliances tree, select the appliance you want to administer using the REST interface and click **User Interface**.
3. Under **UI Access**, select **Enable REST interface over HTTP** or **Enable REST interface over HTTPS** as needed.
4. Click **Save Changes**.

Give permission to access the interface

You must add permission to access the REST interface to an administrator role for those users who are supposed to work with the interface.

Task

1. Select **Accounts** | **Administrator Accounts**.
2. In the **Roles** area, select an administrator role and click **Edit**.

   The **Edit Role** window opens.
3. Select **REST interface accessible**.
4. Click **OK** to close the window.
5. Click **Save Changes**.

You can now assign this administrator role to the appropriate users.

Instead of adding access permission to an existing role, you can also create a new role with this permission and name it, for example, `REST Admin`.

Working with the REST interface

When working with the REST interface that is provided for a Web Gateway appliance, you send requests to this interface to have particular activities completed.

You can send single HTTP or HTTPS requests that are immediately processed or run these requests in a script, for example, in a bash script. The latter is the typical use.

You can also send requests for completing activities on other appliances that are connected as nodes in a Central Management cluster to the appliance where you are working

Requests are sent using a client of the appliance, which in turn provides a server for processing the requests and sending responses. You are assigned a particular amount of work space on this server. For some types of changes, you also need to send a commit request.

As this client, you can- use a data transfer tool, for example, *curl* (Client for URLs).

Before you can send requests for completing any activities, you must log on to the REST interface, authenticate and receive a session ID.

The REST interface is provided in a particular format known as the *ATOM* format.

Sample script for sending a request

The following is an example of a bash script that sends a request to the REST interface using curl. The purpose of the request is to create a configuration backup.

The script does basically the following:

- Logs on and authenticates to the REST interface on an appliance
- Sends a request to create a backup file
- Logs off again

The script also uses a variable for the URL that is specified in the request for logging on to the REST interface. The variable is set at the beginning.

When a sample command or a script with commands is shown in this documentation, a command can extend over two or more lines. When working with the REST interface, you must enter any command completely within a single line.

```
#!bin/bash
## Set URL variable for access to REST interface
REST="http://localhost:4711/Konfigurator/REST"
## Log on and authenticate
curl -c cookies.txt -H "Authorization: Basic YWRtaW46d2ViZ2F0ZXdheQ==" -X POST "$REST/login"
## Create backup file
curl -b cookies.txt -X POST "$REST/backup" -o filename.backup
## Log off again
curl -b cookies.txt -X POST "$REST/logout"
```

Using curl as the data transfer tool

To send requests to the REST interface on an appliance, you can use curl as the data transfer tool.

A request sent with curl usually has three main parts: the `curl` command, one or multiple options, and a URL.

For example, in the following backup request, the `curl` command appears with the `-b` option for sending cookies that have been collected in a text file and the `-o` option, which stores the output of the request in another file. The `-X` option is for the request method.

```
curl -b cookies.txt -X POST "$REST/backup" -o filename.backup
```

The URL is specified as a variable that has the IP address, port number, and other information needed for access to the REST interface on an appliance as its value. It is followed by the name of the activity that is to be performed.

Using these and other options of curl together with the appropriate URLs, you can send requests to the REST interface on an appliance to perform activities as needed.

The curl data transfer tool is available under Linux and other UNIX operating systems and described in full detail, for example, on the *curl* man page.

Request methods

The request method is specified in curl by the `-X` option. When working with the REST interface on an appliance, the GET, POST, PUT, and DELETE methods can be used, for example, as follows:

```
curl -X POST <URL>
```

If no request method is specified, GET is the default method.

Headers

When a header is sent with a request, it is specified by the `-H` option, for example, as follows:

```
curl -H " <header name>:<header value>" -X POST <URL>
```

You can send multiple headers within one request, repeating the `-H` option letter before each header.

```
curl -H "<header name 1>:<header value 1>" -H "<header name 2>:<header value 2>"
-X POST <URL>
```

A request normally includes an `Accept` header that has `application/atom+xml` as its value. In curl, `Accept: */*` is sent as a default, which is accepted by the REST interface, so you can leave out this header in many cases.

However, if you send data in the body of a request, you must include the `Content-Type` header with `application/atom+xml` as its value. You must also include the `Content-Length` header and set it correctly. The latter is done in curl by default, so you need not do it explicitly when using this tool.

If you want to include the header of the response that you receive upon a request in its output, you must insert the `-i` option.

```
curl -i -c cookies.txt -H "Authorization: Basic YWRtaW46d2ViZ2F0ZXdheQ=="
-X POST "$REST/login"
```

The `-v` option creates verbose output, which means that the request header is included.

URLs

A URL in a request specifies a protocol, which can be HTTP or HTTPS in communication with the REST interface, the IP address or host name and the port number of the appliance that a request is sent to, and the internal path on the appliance to the REST interface.

This is followed by the name of the activity that should be performed and further parameters if there are any.

As the REST interface is located within the configurator subsystem of an appliance, the internal name of this subsystem, which is *Konfigurator*, appears in the URL.

A URL in, for example, a logon request, could therefore look as follows:

```
curl -X POST "http://localhost:4711/Konfigurator/$REST
/login?userName=myusername&pass=mypassword"
```

In this request, the URL also has query parameters for the logon credentials. Query parameters are introduced by a `?` (question mark) and separated by an `&` (ampersand), as shown. A URL can also have matrix parameters, which are introduced by a `;` (semicolon).

For correct URL encoding, spaces in a URL must be filled with the symbols `%20`. So, for example, `Bob Smith` becomes `Bob%20Smith`.

You can use a variable within a URL for easier code writing and reading. For example, if you have set the `$REST` variable accordingly, the above request could look as follows:

```
curl -X POST "$REST/login?userName=myusername&pass=mypassword"
```

Sending data in the request body

For sending data in the body of a request, the `-d` option is used, followed by the name of the file that contains the data.

```
curl -b cookies.txt -X POST -d @file.txt "$REST/list?name=newlist&type=string"
```

If you are sending only binary data, the option to use is `– – data-binary`.

```
curl -b cookies.txt --data-binary @file.backup -X POST "$REST/restore"
-H "Content-Type: text/plain; charset=UTF-8"
```

You can use the `@` symbol after the option name to indicate a file name.

Authenticating to the interface

Before you can use the REST interface to perform any activities on an appliance you need to authenticate.

To authenticate, you submit user name and password in the logon request that you send to the REST interface.

There are two ways to submit them:

- Using query parameters
- Using an authentication header

After a successful authentication, the response contains the session ID, which you must include in each of your following requests.

Using query parameters for authentication

You can submit your credentials with query parameters that you add to the URL in your logon request.

```
curl -i -X POST "$REST/login?userName=myusername&pass=mypassword"
```

Using an authentication header

You can also use the Basis Access Authentication method to authenticate, which requires that you submit your credentials in an authentication header.

```
curl -i -H "Authorization: Basic YWRtaW46d2ViZ2F0ZXdheQ==" -X POST "$REST/login"
```

In the authentication header, the string after `Authorization: Basic` is the Base64-encoded representation of your user name and password.

Session ID

The session ID is sent to you in the response to your logon request. A session ID looks, for example, like this:

`D0EFF1F50909466159728F28465CF763`

It is either contained in the response body:

```
<entryxmlns="http://www.w3.org/2005/Atom">
<contenttype="text">D0EFF1F50909466159728F28465CF763</content></entry>
```

or in a Set-Cookie header:

```
Set-Cookie: JSESSIONID=D0EFF1F50909466159728F28465CF763
```

In the requests of the sessions that follow the logon request, you must include the session ID as `JSESSIONID`.

For easier code writing and reading, you can set a variable to the value of the ID and use it for including the ID.

```
export SESSIONID=D0EFF1F50909466159728F28465CF763
```

You can append the ID as a matrix parameter to the URL, preceded by a semicolon.

```
curl -i "$REST/appliances;jsessionid=$SESSIONID"
```

Alternatively, you can send the ID in a Cookie header.

```
curl -i -H "Cookie: JSESSIONID=$SESSIONID" "$REST/appliances"
```

The `-c` option in curl allows you to collect all cookies in a text file, which is then sent with subsequent requests.

```
curl -i -c cookies.txt -H "Authorization: Basic YWRtaW46d2ViZ2F0ZXdheQ=="
-X POST "$REST/login"
```

For sending a cookie file with a request, the `-b` option is used:

```
curl -i -b cookies.txt "$REST/appliances"
```

Requesting resources

A request sent to the REST interface regarding system files, log files, lists, and some other items is considered to be a request for resources.

The response to a request for resources can be one of the following:

- **Entry** — An entry delivers information in xml format about an individual resource, such as its ID, name, or the URL that can be used to access it
- **Feed** — A feed delivers information in xml format about a collection of resources.

  A feed can, for example, be a list of appliances that are available as nodes in a Central Management configuration, or a list of all lists that exist on an appliance, or a list of all lists of a particular type.
- **Binary data** — Binary data is delivered in a file that you requested for downloading.

A response can also be empty. This is the case when the requested data is not available.

Reducing xml data overhead

You can reduce the xml data overhead that you receive with a response, by including an appropriate `Accept` header in a request for resources. For this purpose, the header value must be `application/mwg+xml`.

Instead of an entry in the normal Atom format, you will then receive only the xml data from the content part of that format.

Instead of a feed in Atom format, you will only receive a list of IDs for the resources you asked for.

Similarly, you can reduce xml data overhead when working with the resources, for example, when modifying them. For this, you need to set the `Content-Type` header to `application/mwg+xml`.

Paging a feed

When requesting a feed, you can use paging, which means you can ask for a feed that is divided into pages.

Paging information is specified by query parameters that are added to the URL in a request. The following two parameters can be used:

- **pageSize** — Maximum number of elements on a page
- **page** — Page number

A request for a feed that uses paging could look as follows:

```
curl -i -b cookies.txt "$REST/list?pageSize=10&page=4"
```

If a feed is, for example, a list of 35 lists, the `pageSize` parameter in the above request divides it up into four pages, three of which contain ten lists, while the last one contains only five. The last page is also the one that is delivered.

Navigating within a feed

To allow navigation within a feed, the xml file that you receive contains appropriate links.

Using these links, you can go to the current, next, previous, first, and last page, respectively.

Performing basic REST operations

When working with the REST interface, you can perform several basic operations such as logging on and off, committing changes, and creating a configuration backup.

Use a POST request for these operations and specify each operation by a parameter that you add to the URL of the request.

For example, this is a request to log off from the REST interface on an appliance, using a curl command:

When a sample command or script with commands is shown in this documentation, a command can extend over two or more lines. When working with the REST interface, you must enter any command completely within a single line.

```
curl -i -b cookies.txt -X POST "$REST/logout"
```

Parameters for basic REST operations include:

- `login` — Log on
- `logout` — Log off
- `heartbeat` — Keep a session alive
- `commit` — Commit changes
- `discard` — Discard changes
- `backup` — Back up the configuration
- `restore` — Restore the configuration
- `updateEngines` — Update the filter engines

Logging on

Request format:

URL/login?userName=<user name>&pass=<password>

To log on to the REST interface on an appliance, use the `login` parameter. Within the request, you also submit your credentials for authentication. If authentication is successful, the response to the logon request provides a session ID.

Sample command:

```
curl -i -X POST "$REST/login?userName=myusername&pass=mypassword"
```

Request parameters:

Parameter	Type	Description
userName	String	User name submitted for authenticating to the REST interface Default: None
pass	String	Password submitted for authenticating to the REST interface Default: None

Logging off

Request format:

URL/logout

To log off from the REST interface on an appliance, use the `logout` parameter. Logging off deletes the session information and discards the changes made in a session that have not been committed.

Sample command:

```
curl -i -X POST "$REST/logout"
```

Request parameters:

None

Keeping a session alive

Request format:

URL/heartbeat

Using the `heartbeat` parameter in a request keeps the current session alive.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/heartbeat"
```

Request parameters:

None

Committing changes

Request format:

URL/commit

To commit changes that have been made to items such as system files, log files, and lists on an appliance, use the `commit` parameter.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/commit"
```

Request parameters:

None

Discarding changes

Request format:

URL/discard

To discard changes that have been made to items such as system files, log files, and lists on an appliance, use the `discard` parameter.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/discard"
```

Request parameters:

None

Backing up the configuration

Request format:

URL/backup or **URL/backup?password=string** or **URL/backup?backUpMAS=boolean&password=string**

To create a configuration backup for the appliance where you are currently working, use the `backup` parameter. When backing up or restoring a configuration, no response header is required as part of the output, so the `-i` option is not included in the command for performing the request.

The configuration backup is stored in the file that is specified as output in the command.

Sample commands:

```
curl -b cookies.txt -X POST "$REST/backup" -o filename.backup
or:
curl -b cookies.txt -X POST "$REST/backup?password=yourpassword" -o filename.backup
or:
curl -b cookies.txt -X POST "$REST/backup?backUpMAS=true&password=yourpassword" -o
filename.backup
```

Request parameters:

Parameter	Type	Description
backUpMAS	Boolean	If true, a password is used to encrypt the backup. Default: false
password	String	Password used to encrypt the backup. If no password is specified, the backup is not encrypted. Default: None

Restoring the configuration

Request format:

URL/restore?fullrestore=boolean&restoreMAS=boolean&password=string

To restore the configuration of the appliance you are currently working on, use the `restore` parameter. You must also specify a Content-Type header for the type of the backup file.

Sample command:

```
curl -b cookies.txt --data-binary @filename.backup -X POST "$REST/restore" -H "Content-Type:
text/plain;charset=UTF-8"
```

Request parameters:

Parameter	Type	Description
fullrestore	Boolean	If true, a configuration is restored completely. If false, only the policy is restored, omitting the system settings for an appliance. Default: false
restoreMAS	Boolean	If true, MAS is also restored if available. Default: false
password	String	Password used to decrypt the backup. If no password is specified, the backup is not decrypted.

Updating the filter engines

Request format:

URL/updateEngines

To perform an update of the filter engines on an appliance with data that is provided offline, use the *updateEngines* parameter. You must also specify a Content-Type header for the type of the update file.

Sample command:

```
curl -b cookies.txt --data-binary @mwg7-linux-mix-small.upd -X POST "$REST/updateEngines"
 -H "Content-Type: text/plain;charset=UTF-8"
```

Request parameters:

None

Sample script for performing basic REST operations

The following bash script performs several basic operations: logging on and authenticating to the REST interface on an appliance, creating a backup file, and logging off again.

Before these operations are performed, the script sets a URL variable for accessing the REST interface.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Create backup file
curl -b cookies.txt -X POST "$REST/backup" -o filename.backup
## Log off again
curl -b cookies.txt -X POST "$REST/logout"
```

Requesting version information

You can request the version of the REST interface for the Web Gateway appliance where you are currently working, as well as the version of the standard user interface.

You can request both versions at once or each version separately.

Requesting the interface versions

Request format:

URL/version

To request version information, use the `version` parameter. When used without further specification, the response to this request is a feed with the version numbers of both interfaces in XML format.

Sample commands:

```
curl -i -b cookies.txt .X GET "$REST/version"
```

Request parameters:

None

Requesting the version of a particular interface

Request formats:

URL/version/mwg-rest

URL/version/mwg-ui

To request version information for a particular interface, the interface name is added to the *version* parameter. The response to this request is a feed with the respective version number in XML format.

Sample commands:

```
 curl -i -b cookies.text .X GET "$REST/version/mwg-rest"
curl -i -b cookies.text .X GET "$REST/version/mwg-ui"
```

Request parameters:

Parameter	Type	Description
name	String	Name of the interface that version information is requested for Default: None

Sample script for requesting version information

The following bash script requests version information for both the REST interface and the standard user interface of the appliance where you are currently working.

Before specifying this request, the script sets a URL variable for accessing the REST interface.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -c cookies.txt -H "Authorization: Basic YWRtaW46d2ViZ2F0ZXdheQ==" -X POST "$REST/login"
## Request version information
curl -b cookies.txt -X GET "$REST/version"
## Log off again
curl -b cookies.txt -X POST "$REST/logout"
```

Working on appliances

After logging on to the REST interface on one appliance, you can complete activities on any other appliance that is connected. You can also complete some activities on all appliances at once.
An activity that is completed on all appliances at once is, for example, to import a license for all of them.

Individual appliances are identified in requests by their *UUIDs (Universal Unique Identifiers)*

The UUID of an appliance looks like this:

081EEDBC-7978-4611-9B96-CB388EEFC4BC

To find out about the UUID of an appliance, you can request a feed of all appliances that are connected as nodes in a Central Management cluster to the one where you are currently working.

The feed that is sent in response to your request includes a list of the UUIDs for all nodes. You can then identify an appliance by its UUID and, for example, shut down this appliance.

Importing a license

Request format:

<URL>/appliances/license

To import a license onto the appliances that are connected in a Central Management cluster, use the `appliances` and `license` parameters in a request.

You also need to specify a Content-Type header for the type of the license file.

Sample command:

```
curl -i -b cookies.txt " -H "Content-Type: text/plain; charset=UTF-8" -X POST "$REST/
appliances/license" --data-binary @license.xml
```

Requesting a feed with UUIDs for all appliances

Request format:

<URL>/appliances

To request a feed with a list of UUIDs for all appliances that run as nodes in a cluster, use the `appliances` parameter.

Sample command:

```
curl -i -b cookies.txt -X GET "$REST/appliances"
```

Request parameters:

None

Requesting a page in a feed for all appliances

Request format:

<URL>/appliances?page=<int>&pageSize=<int>

To request a particular page in a feed for all appliances that run as nodes in a cluster, use the *appliances* parameter with a page parameter appended. You can also append another parameter to request a particular page size.

Sample command:

```
curl -i -b cookies.txt -X GET "$REST/appliances?page=3&pageSize=2"
```

Request parameters:

Parameter	Type	Description
page	Integer	Number of the page in a feed for all appliances Default: 1
pageSize	Integer	Size of the page in a feed for all appliances Default: –1

Actions

Actions that are performed on individual appliances are specified in a request by their names, which are preceded by the `action` parameter.

These actions do not involve a modification of resources and are performed instantly, which means no request to commit them is required.

You can perform the following actions:

- `restart` — Restart an appliance
- `shutdown` — Shut down an appliance
- `flushcache` — Flush the cache
- `rotateLogs` — Rotate log files
- `rotateAndPushLog`s — Rotate and push log files
- `license` — Import a license

Restarting an appliance

Request format:

<URL>/<UUID>/action/restart

To restart an appliance, use `restart` as the action name in a request.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/action/restart"
```

Request parameters:

Parameter	Type	Description
UUID	String	UUID of the appliance that you want to restart Default: None

Shutting down an appliance

Request format:

<URL>/<UUID>/action/shutdown

To shut down an appliance, use `shutdown` as the action name in a request.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/action/shutdown"
```

Request parameters:

Parameter	Type	Description
UUID	String	UUID of the appliance that you want to shutdown Default: None

Flushing the cache

Request format:

<URL>/<UUID>/action/flushcache

To flush the cache on an appliance, use `flushcache` as the action name in a request.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/action/flushcache"
```

Request parameters:

Parameter	Type	Description
UUID	String	UUID of the appliance where you want to flush the cache Default: None

Rotating log files

Request format:

<URL>/<UUID>/action/rotateLogs

To rotate the log files on an appliance, use `rotateLogs` as the action name in a request.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/action/rotateLogs"
```

Request parameters:

Parameter	Type	Description
UUID	String	UUID of the appliance where you want to rotate log files Default: None

Rotating and pushing log files

Request format:

<URL>/<UUID>/action/rotateAndPushLogs

To rotate the log files on an appliance and push them to a remote server, use *rotateAndPushLogs* as the action name in a request.

Sample command:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/action/rotateAndPushLogs"
```

Request parameters:

Parameter	Type	Description
UUID	String	UUID of the appliance where you want to rotate and push log files Default: None

Sample script for working on individual appliances

The following bash script rotates logs as an example for completing an action on an individual appliance.

Before this action is completed, the script sets a URL variable for accessing the REST interface.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Rotate log files
curl -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/action/rotateLogs"
## Log off again
curl -b cookies.txt -X POST "$REST/logout"
```

Working with configurations and settings

Using the REST interface, you can work with the configurations that were created for Web Gateway individual appliances and appliance clusters. You can also work with the settings for the filter modules on the appliances.

Identifying configurations and settings

When working with configuration and settings, you must identify them in the commands to complete particular activities.

The `cfgID` parameter serves as an identifier. Its value is usually the name of a file that contains an xml representation of a configuration or settings, for example, `com.scur.engine.coaching.configuration`.

You can look up the identifiers in the feed that you receive when retrieving a configuration or settings. In the entries of the feed, identifiers are tagged differently for configurations and settings.

- **Identifier for a configuration** — Last string of the section tagged as `<id>` in an entry, immediately following the `/cfg/` section.

  For example, `com.scur.engine.cmclusternode.configuration` is the identifier for a cluster configuration in this entry:

  ```
  <entry><id>7284F2DE-BE26-0CC9-E825-000000468CBE/cfg/com.scur.engine.cmclusternode
  .configuration</id><title>Central Management</title><type>com.scur.engine.cmclusternode</
  type>
  <link href="http://localhost:4711/Konfigurator/REST/appliances/7284F2DE-BE26-0CC9
  -E825-000000468CBE/configuration/com.scur.engine.cmclusternode.configuration"
  rel="self"/></entry>
  ```

  The `<title>` section serves as an alternative identifier. In the above example, `com.scur.engine.cmclusternode` also identifies the cluster configuration.

- **Identifier for settings** — Tagged as `<id>` in an entry.

  For example, `com.scur.mainaction.block.11376` is an identifier for settings in this entry:

  ```
  <entry><id>com.scur.mainaction.block.11376</id><title>Unknown Certificate
  Authorities</title><type>com.scur.mainaction.block</type><link href="http://localhost:4711
  Konfigurator/REST/setting/com.scur.mainaction.block.11376" rel="self"/></entry>
  ```

  The filter module that the settings belong to is tagged as `<title>`.

  In this example, the `com.scur.mainaction.block.11376` settings are configured for the module that filters unknown certificate authorities.

Retrieving a configuration or settings

Request formats

<URL>/appliances/<UUID>/configuration

<URL>/appliances/<UUID>/configuration?page=<int>&pageSize=<int>

<URL>/appliances/<UUID>/configuration?type=<string>&name=<string>

<URL>/appliances/<UUID>/configuration?type=<string>&name=<string>&page=<int>&pageSize=<int>

<URL>/cluster/configuration

<URL>/cluster/configuration?type=<string>&name=<string>&page=<int>&pageSize=<int>

<URL>/setting

<URL>/setting?type=<string>&name=<string>

You can retrieve these configurations and settings:

- Configuration for an appliance that is connected to the appliance where you are currently working
- Configuration for a cluster of appliances including the appliance where you are currently working
- Settings of the appliance where you are currently working

You can also retrieve configurations and settings for filter modules. You can retrieve all configurations and settings that exist for a module or only particular configurations and settings.

The response to this request is a feed with the configuration or settings in xml format. You can request a particular page of the feed and specify the page size.

Sample commands

Retrieve the configuration of an appliance:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration"
```

Retrieve a cluster configuration:

```
curl -i -b cookies.txt -X GET "$REST/cluster/configuration"
```

Retrieve the settings of the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/setting"
```

Retrieve all settings that exist for a filter:

```
curl -i -b cookies.txt -X GET "$REST/setting?type=com.scur.engine.antivirus"
```

Retrieve particular filter settings:

```
curl -i -b cookies.txt -X GET "$REST
/setting?type=com.scur.engine.antivirus&name=Gateway%20Anti-Malware"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
page	Integer	Number of a page in a feed with a configuration or settings Default: 1
pageSize	Integer	Size of a page in a feed Default: –1
type	String	Name of a filter module on an appliance, for example, `com.scur.engine.antivirus` You can look up the names of the filter modules in the feed that you receive when retrieving a configuration or settings. In this feed, filter names are tagged as `<type>`. Default: None
name	String	Name of particular settings for a filter module You can look up the settings names on the standard user interface of Web Gateway or in the feed that you receive when retrieving a configuration or settings. In this feed, settings names are tagged as `<title>`. Spaces in the settings names must be filled with `%20`, for example, `Gateway %20Anti-Malware` Default: None

Retrieving an xml representation

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>

<URL>/cluster/configuration/<cfgID>

<URL>/setting/<cfgID>

You can retrieve a file in xml format that has been created on a Web Gateway appliance to represent a configuration or settings.

When retrieving an xml representation, you use an identifying string, also known as *cfgID*, to identify a configuration or settings.

The response to this request includes the xml representation in the response body.

Sample commands

Retrieve the xml representation of a configuration for an appliance:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration/com.scur.engine.coaching.configuration"
```

Retrieve the xml representation of a cluster configuration:

```
curl -i -b cookies.txt -X GET "$REST/cluster/configuration
/com.scur.cm_cluster_global.internal.configuration"
```

Retrieve the xml representation of the settings for the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/setting/com.scur.mainaction.block.11376"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None

Modifying a configuration or settings

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>

<URL>/cluster/configuration/<cfgID>

<URL>/setting/<cfgID>

You can modify a configuration or settings by applying changes to the file that contains an xml representation of them.

When specifying the changes, you provide the name of the file that is to be changed and the name of the file that contains the changes. You must also specify a Content-Type header.

After specifying the changes, you must commit them using a separate command.

The response to this request includes the modified configuration or settings in the response body.

Sample commands

Modify the xml representation of a configuration for an appliance:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d
@changesToModifyConfiguration.xml
-X PUT "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration/com.scur.engine.coaching.configuration"
```

Modify the xml representation of a cluster configuration:

```
curl -i -b cookies.txt -X PUT -d @changesToModifyClusterConfiguration.xml
-H "Content-Type: application/xml" "$REST/cluster/configuration
/com.scur.cm_cluster_global.internal.configuration"
```

Modify the xml representation of the settings for the appliance where you are currently working:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @changesToModifySettings.xml
-X PUT "$REST/setting/com.scur.mainaction.block.11376"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None

Retrieving a list of properties

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>/property

<URL>/appliances/<UUID>/configuration/<cfgID>/property?page=<int>&pageSize=<int>

<URL>/cluster/configuration/<cfgID>/property

<URL>/cluster/configuration/<cfgID>/property?page=<int>&pageSize=<int>

<URL>/setting/<cfgID>/property

<URL>/setting/<cfgID>/property?page=<int>&pageSize=<int>

You can retrieve a list with the properties of a configuration or settings.

When retrieving this list, you provide the name of the file with the xml representation that includes the properties.

The response to this request is a feed that includes the list in xml format. You can retrieve a particular page of this feed and specify the page size.

Sample commands

Retrieve a list of the properties in the configuration for an appliance:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration/com.scur.engine.coaching.configuration/property"
```

Retrieve a list of the properties in a cluster configuration:

```
curl -i -b cookies.txt -X GET "$REST/cluster/configuration
/com.scur.cm_cluster_global.internal.configuration/property"
```

Retrieve a list of the properties in the settings of the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/setting/com.scur.mainaction.block.11376/property"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None
page	Integer	Number of a page in a feed for a configuration or settings Default: 1
pageSize	Integer	Size of a page in a feed Default: -1

Retrieving a property

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>/property/<propertyname>

<URL>/cluster/configuration/<cfgID>/property/<propertyname>

<URL>/setting/<cfgID>/property/<propertyname>

You can retrieve a property from a configuration or settings.

When retrieving this property, you provide its name and the name of the file with the xml representation that includes the property.

The response to this request includes the property in xml format.

Sample commands

Retrieve a property from a configuration for an appliance:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration/com.scur.engine.coaching.configuration/property/sendsync"
```

Retrieve a property from a cluster configuration:

```
curl -i -b cookies.txt -X GET "$REST/cluster/configuration
/com.scur.cm_cluster_global.internal.configuration/property/
DataUsageStatementVersionAccepted"
```

Retrieve a property from the settings of the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/setting/com.scur.mainaction.block.11376/property
/TemplateName"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None
propertyname	String	Name of a property in a configuration or settings Default: None

Modifying a property value

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>/property/<propertyname>

<URL>/cluster/configuration/<cfgID>/property/<propertyname>

<URL>/setting/<cfgID>/property/<propertyname>

You can modify the value of a property in a configuration or settings.

When modifying this value, you provide the name of the file with the property that has its value changed and the name of the file with the new value. You must also specify a Content-Type header.

After modifying the value, you must commit this change using a separate command.

The response to this request includes the modified property value in xml format.

Sample commands

Modify the value of a property in a configuration for an appliance:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @changesToModifyProperty.xml
-X PUT "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/configuration
/com.scur.engine.coaching.configuration/property/sendsync"
```

Modify the value of a property in a cluster configuration:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @changesToModifyProperty.xml
-X PUT "$REST/cluster/configuration/com.scur.cm_cluster_global.internal.configuration
/property/DataUsageStatementVersionAccepted"
```

Modify the value of a property in the settings of the appliance where you are currently working:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @changesToModifyProperty.xml
-X PUT "$REST/setting/com.scur.mainaction.block.11376/property/TemplateName"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None
propertyname	String	Name of a property in a configuration or settings Default: None

Retrieving the default value of a property

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>/property/<propertyname>/default

<URL>/cluster/configuration/<cfgID>/property/<propertyname>/default

<URL>/setting/<cfgID>/property/<propertyname>/default

You can retrieve the default value of a property from a configuration or settings.

When retrieving this value, you provide the property name and the name of the file with the xml representation that includes the property.

The response to this request includes the default value in xml format.

Sample commands

Retrieve the default value of an individual property in a configuration for an individual appliance.

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/
configuration
/com.scur.engine.coaching.configuration/property/sendsync/default"
```

Retrieve the default value of a property in a cluster configuration.

```
curl -i -b cookies.txt -X GET "$REST/cluster/configuration
/com.scur.cm_cluster_global.internal.configuration/property/DataUsageStatementVersionAccepted
/default"
```

Retrieve the default value of a property in the settings of the appliance where you are currently working.

```
curl -i -b cookies.txt -X GET "$REST/setting/com.scur.mainaction.block.11376/property
/TemplateName/default"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None
propertyname	String	Name of a property in a configuration or settings Default: None

Setting a property to its default value

Request formats

<URL>/appliances/<UUID>/configuration/<cfgID>/property/<propertyname>/default

<URL>/cluster/configuration/<cfgID>/property/<propertyname>/default

<URL>/setting/<cfgID>/property/<propertyname>/default

You can set a property to its default value in a configuration or settings.

When setting this value, you provide the property name and the name of the file with the xml representation that includes the property.

The response to this request includes the property set to its default value in xml format.

Sample commands

Set a property to its default value in a configuration for an appliance:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration/com.scur.engine.coaching.configuration/property/sendsync/default"
```

Set a property to its default value in a cluster configuration:

```
curl -i -b cookies.txt -X POST "$REST/cluster/configuration
/com.scur.cm_cluster_global.internal.configuration
/property/DataUsageStatementVersionAccepted/default"
```

Set a property to its default value in the settings of the appliance where you are currently working:

```
curl -i -b cookies.txt -X POST "$REST/setting/com.scur.mainaction.block.11376/property
/TemplateName/default"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None
propertyname	String	Name of a property in a configuration or settings Default: None

Retrieving a list with the actions of a configuration

Request format

<URL>/appliances/<UUID>/configuration/<cfgID>/action

You can retrieve a list with the actions that are executed according to the rules of a configuration for an appliance.

When retrieving this list, you provide the name of the file with the xml representation that includes the actions.

The response to this request is a feed that includes the list in xml format.

Sample command

Retrieve a list with the actions in a configuration for an appliance:

```
curl -i -b cookies.txt -X GET "$REST/appliances/7284F2DE-BE26-0CC9-E825-0000004A637C
/configuration/com.scur.engine.cmclusternode.configuration/action"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an individual appliance
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None

Triggering an action

Request format

<URL>/appliances/<UUID>/configuration/<cfgID>/action/*actionname*

You can trigger an action that is included in a configuration for an appliance.

When triggering this action, you provide its name and the name of the file with the xml representation that includes the action.

Sample command

Trigger an action in a configuration for an appliance:

```
curl -i -b cookies.txt -X POST "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/configuration/com.scur.engine.cmclusternode/action/update_engines_all_standalone"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an individual appliance
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None
actionname	String	Name of an action that is included in the xml representation of a configuration or settings Default: None

Deleting settings

Request format

<URL>/setting/<cfgID>

You can delete settings for the appliance where you are currently working.

When deleting these settings, you provide the name of the file with the xml representation that includes these settings.

After deleting the settings, you must commit this change using a separate command.

The response to this request includes the deleted settings in xml format.

Sample command

Delete settings for the appliance where you are currently working:

```
curl -i -b cookies.txt -X DELETE "$REST/setting/com.scur.mainaction.block.11376"
```

Variable request parameters

Parameter	Type	Description
cfgID	String	String used as an identifier for a configuration or settings This string is usually the name of a file that contains an xml representation of the configuration or settings, for example, `com.scur.engine.coaching.configuration`. Default: None

Adding settings with content

Request format

<URL>/setting?type=<string>&name=<string>

You can add new settings for a filter module on the appliance where you are currently working.

When adding these settings, you provide the names of the filter module and the settings.

You must also specify a Content-Type header and append a file in xml format that provides the content of the settings.

After adding the settings, you must commit this change using a separate command.

The response to this request includes the added settings in xml format.

Sample command

Add new settings for a filter module on the appliance where you are currently working:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d
@newSettingwithoutNameAndType.xml
-X POST "$REST/setting?type=com.scur.mainaction.block&name=Malware%20Detected"
```

Variable request parameters

Parameter	Type	Description
type	String	Name of a filter module on an appliance, for example, `com.scur.engine.antivirus` You can look up the names of the filter modules in the feed that you receive when retrieving a configuration or settings. In this feed, filter names are tagged as `<type>`. Default: None
name	String	Name of new settings that are added for a filter module. Spaces in the settings names must be filled with `%20`, for example, `Gateway%20New %20Anti-Malware`. Default: None

Adding settings with content including type and name

Request format

<URL>/setting

You can add new settings for a filter module on the appliance where you are currently working with the names of the filter module and the settings already included in the settings content.

When adding settings in this way, you do not provide the names of the filter module and the settings as request parameters. You specify a Content-Type header and append a file in xml format that provides the content of the settings.

After adding the settings, you must commit this change using a separate command.

The response to this request includes the added filter settings in xml format.

Sample command

Add new settings for a filter module on the appliance where you are currently working with content including the filter and settings names:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @newSettingWithNameAndType.xml
-X POST "$REST/setting"
```

Variable request parameters

None

Adding new default settings

Request format

<URL>/setting?type=<string>&name=<string>

You can add new settings for a filter module on the appliance where you are currently working with default values configured for all settings options.

When adding these settings, you provide the name of the filter module and a name for the new default settings.

After adding the settings, you must commit this change using a separate command.

The response to this request includes the new settings with their default values in xml format.

Sample command

Add new settings with default values for a filter module on the appliance where you are currently working:

```
curl -i -b cookies.txt -X POST "$REST
/setting?type=com.scur.mainaction.block&name=New%20Blocking"
```

Variable request parameters

Parameter	Type	Description
type	String	Name of a filter module on an appliance, for example, `com.scur.engine.antivirus` You can look up the names of the filter modules in the feed that you receive when retrieving a configuration or settings. In this feed, filter names are tagged as `<type>`. Default: None
name	String	Name of new default settings that are added for a filter module Spaces in the settings names must be filled with `%20`, for example, `Gateway%20New %20Default%20Anti-Malware`. Default: None

Sample script for working with configurations and settings

The following bash script modifies the value of a property in a configuration for an appliance.

Before performing this operation, the script sets a URL variable for accessing the REST interface.

Some of the operations performed with configurations and settings require running still another command to commit a change that you requested, for example, modifying a property value. When a commit is required, this is mentioned in the description of the operation.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Modify property value
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @changesToModifyProperty.xml
-X PUT "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/configuration
/com.scur.engine.coaching.configuration/property/sendsync"
## Commit modification
curl -b cookies.txt -X POST "$REST/commit"
## Log off
curl -b cookies.txt -X POST "$REST/logout"
```

Working with system files

You can use the REST interface to work with system files on any appliance in a cluster that includes the appliance where you are currently working.

Modifying system files inadequately can impact the proper operation of an appliance.

With system files, you can:

- Retrieve a list of system files
- Download a system file
- Modify a system file

When running an appliance in FIPS-compliant mode, you cannot modify system files.

Retrieving a list of system files

Request formats

<URL>/appliances/<UUID>/system

<URL>/appliances/<UUID>/system?page=<int>&pageSize=<int>

You can retrieve a list of system files from any appliance in a cluster that includes the appliance where you are currently working.

When retrieving a list of system files from an appliance, you append the `appliances` parameter, the appliance ID, and the **system** parameter.

You can request a particular page of the feed and specify the page size.

In response to your request, you receive an xml file with all system files listed.

Sample command

Retrieve a list of system files from an appliance in a cluster that includes the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/system"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
page	Integer	Number of a page in a feed with a list of system files Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Downloading a system file

Request format

<URL>/appliances/<UUID>/system/<file name>

<URL>/appliances/<UUID>/system/<path>/<file name>

You can download a system file from an appliance in a cluster that includes the appliance where you are currently working.

When downloading a system file from an appliance, you append the `appliances` parameter and the appliance ID, as well as the path to the system file if there is any and the file name. You must also specify an Accept header.

Using the `-o` parameter, you can store the downloaded system file as a local file under its name on the appliance where you downloaded it from.

Sample command

Download a system file from an appliance in a cluster that includes the appliance where you are currently working.

```
curl -b cookies.txt -H "Accept: application/x-download"
 -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/system/hosts" -o hosts.txt
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
<path>	String	Path to a system file that is downloaded Default: None
<file name>	String	Name of a system file that is downloaded Default: None

Modifying a system file

Request format

<URL>/appliances/<UUID>/system/<file name>

<URL>/appliances/<UUID>/system/<path>/<file name>

You can modify a system file on an appliance in a cluster that includes the appliance where you are currently working.

When running an appliance in FIPS-compliant mode, you cannot modify system files.

When modifying a system file on an appliance, you append the **appliances** parameter and the appliance ID, as well as the path to the system file if there is any and the file name.

You must also specify a Content-Type header and append a file with the modified data in binary format.

After modifying the system file, you must commit this change using a separate command.

Sample command

Modify a system file on an appliance in a cluster that includes the appliance where you are currently working.

```
curl -b cookies.txt -H "Content-Type: */*" -X PUT "$REST/appliances
/081EEDBC-7978-4611-9B96-CB388EEFC4BC/system/hosts" -d @hosts.txt
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
<path>	String	Path to a system file that is modified Default: None
<file name>	String	Name of a system file that is modified Default: None

Sample script for working with system files

The following bash script modifies a system file on an appliance in a cluster that includes the appliance where you are currently working.

Before performing this operation, the script sets a URL variable for accessing the REST interface.

Modifying a system file requires running a separate command to commit the modification. Other operations, for example, retrieving a list of system files, do not require a commit.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Modify system file
curl -i -b cookies.txt -H "Content-Type: */* -X PUT "$REST/appliances
/081EEDBC-7978-4611-9B96-CB388EEFC4BC/system/hosts" -d @hosts.txt
## Commit modification
curl -b cookies.txt -X POST "$REST/commit"
## Log off
curl -b cookies.txt -X POST "$REST/logout"
```

Working with log files

You can use the REST interface to work with log files on any appliance in a cluster that includes the appliance where you are currently working.

When working with log files, you include the `appliances` parameter and identify an appliance in all requests.

You also append the `log` parameter and other parameters as needed to complete particular activities with log files.

With log files, you can:

- Retrieve a list of log files
- Download a log file
- Delete a log file

Retrieving a list of log files

Request formats

<URL>/appliances/<UUID>/log

<URL>/appliances/<UUID>/log?page=<int>&pageSize=<int>

You can retrieve a list of log files on an appliance in a cluster that includes the appliance where you are currently working.

You can request a particular page of the feed that you receive in response and specify the page size.

The feed provides MIME type information in XML format, indicating for every list item whether it is a log file or a directory.

- *"application/x-download"* — log file
- *"application/atom+xml; type=feed"* — directory

Sample command

Retrieve a list of log files on an appliance in a cluster that includes the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/log"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
page	Integer	Number of a page in a feed with a list of log files Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Downloading a log file

Request format

<URL>/appliances/<UUID>/log/<subresources>

You can download a log file from an appliance in a cluster that includes the appliance where you are currently working.

When downloading a log file, you provide its path and name. You must also specify an Accept header.

Using the `-O` parameter, you can store the downloaded log file as a local file under its name on the appliance where you downloaded it from.

Sample command

Download a log file from an appliance in a cluster that includes the appliance where you are currently working.

```
curl -b cookies.txt -H "Accept: application/x-download"
 -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC/log/debug/debug_1234.log" -O
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
subresources	String	Path to a file that is downloaded and file name Default: None

Deleting a log file

Request format

<URL>/appliances/<UUID>/log/<subresources>

You can delete a log file on an appliance in a cluster that includes the appliance where you are currently working.

When deleting a log file, you provide its path and name.

Sample command

Delete a log file on an appliance in a cluster that includes the appliance where you are currently working.

```
curl -i -b cookies.txt -X DELETE "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/log/debug/debug_1234.log"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
subresources	String	Path to a file that is deleted and file name Default: None

Sample script for working with log files

The following bash script deletes a log file on an appliance in a cluster that includes the appliance where you are currently working.

Before performing this operation, the script sets a URL variable for accessing the REST interface.

Log file operations do not require a commit.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Delete log file
curl -i -b cookies.txt -X DELETE "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/log/debug/debug_1234.log"
## Log off
curl -b cookies.txt -X POST "$REST/logout"
```

Working with files uploaded for troubleshooting

You can use the REST interface to work with files that have been uploaded to an appliance for troubleshooting.

On the standard Web Gateway interface, you can perform this upload using the **Troubleshooting** top-level menu.

When working with uploaded files on the REST interface, you identify an appliance and append the `files` parameter.

With files uploaded for troubleshooting, you can:

- Retrieve a list of uploaded files
- Download an uploaded file
- Add a file to the uploaded files
- Modify an uploaded file
- Delete an uploaded file

Retrieving a list of uploaded files

Request formats

<URL>/appliances/<UUID>/files

<URL>/appliances/<UUID>/files?page=<int>&pageSize=<int>

You can retrieve a list of uploaded files on an appliance in a cluster that includes the appliance where you are currently working.

You can request a particular page of the feed that contains the list and specify the page size.

Sample commands

Retrieve a list of uploaded files from an appliance:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/files"
```

Request a particular page of the feed with the list and specify the page size:

```
curl -i -b cookies.txt -X GET "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/files?page=4&pageSize=1"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
page	Integer	Number of a page in a feed with uploaded files Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Downloading an uploaded file

Request format

<URL>/appliances/<UUID>/files/<filename>

You can download an uploaded file from an appliance in a cluster that includes the appliance where you are currently working.

When downloading an uploaded file, you specify an Accept header. Using the `–O` parameter, you can store the downloaded data in a local file under its name on the appliance where you downloaded it from.

Sample command

Download an uploaded file from an appliance:

```
curl -b cookies.txt -H "Accept: application/x-download" -X GET "$REST/appliances
/081EEDBC-7978-4611-9B96-CB388EEFC4BC/files/troubleshooting.zip" -O
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
filename	String	Name of an uploaded file Default: None

Adding a file to the uploaded files

Request format

<URL>/appliances/<UUID>/files/<filename>

You can add a file to the uploaded files on an appliance in a cluster that includes the appliance where you are currently working.

When adding a file, you provide its name and append the file in binary format as the request body.

You must also specify a Content-Type header. Do not specify `application/x-www-form-urlencoded`, as the curl tool already appends this type.

Sample command

Add a file to the uploaded files on an appliance:

```
curl -i -b cookies.txt -H "Content-Type: */*" -X POST "$REST/appliances
/081EEDBC-7978-4611-9B96-CB388EEFC4BC/files/moreTroubleshooting.zip"
--data-binary @moreTroubleshooting.zip
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
filename	String	Name of a file that is added to the uploaded files on an appliance Default: None

Modifying an uploaded file

Request format

<URL>/appliances/<UUID>/files/<filename>

You can modify an uploaded file on an appliance in a cluster that includes the appliance where you are currently working.

When modifying an uploaded file, you provide its name and append the file it in binary format with the data required for the modification as the request body.

You must also specify a Content-Type header. Do not specify `application/x-www-form-urlencoded`, as the curl tool already appends this type.

Sample command

Modify an uploaded file on an appliance:

```
curl -i -b cookies.txt -H "Content-Type: */*" -X PUT "$REST/appliances
/081EEDBC-7978-4611-9B96-CB388EEFC4BC/files/moreTroubleshooting.zip"
--data-binary @modificationData.zip
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
filename	String	Name of a file that is modified Default: None

Deleting an uploaded file

Request format

<URL>/appliances/<UUID>/files/<filename>

You can delete an uploaded file on an appliance in a cluster that includes the appliance where you are currently working.

When deleting an uploaded file, you provide its name.

Sample command

Delete an uploaded file on an appliance:

```
curl -i -b cookies.txt -X DELETE "$REST/appliances/081EEDBC-7978-4611-9B96-CB388EEFC4BC
/files/troubleshooting.zip"
```

Variable request parameters

Parameter	Type	Description
UUID	String	Universally unique identifier for an appliance Default: None
filename	String	Name of a file that is deleted Default: None

Sample script for working with uploaded files

The following bash script modifies an uploaded file on an appliance.

Before performing this operation, the script sets a URL variable for accessing the REST interface.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Modify uploaded file
curl -i -b cookies.txt -H "Content-Type:*/*" -X PUT "$REST/appliances
```

```
/081EEDBC-7978-4611-9B96-CB388EEFC4BC/files/moreTroubleshooting.zip"
--data-binary @modificationData.zip
## Log off
curl -b cookies.txt -X POST "$REST/logout"
```

Working with lists

You can use the REST interface to work with lists and their entries on the appliance where you are currently working.

When working with lists, you include the `list` parameter in all requests. For list entries, you add the `entry` parameter.

With lists, you can:

- Retrieve a list of lists
- Retrieve a list
- Add a list with content
- Add a list with content including type and name
- Add an empty list
- Modify a list
- Rename a list
- Copy a list
- Delete a list

With list entries, you can:

- Retrieve a list of list entries
- Retrieve a list entry
- Insert a list entry
- Modify a list entry
- Move a list entry
- Delete a list entry

Retrieving a list of lists

Request formats

<URL>/list

<URL>/list?page=<int>&pageSize=<int>

<URL>/list?type=<string>

<URL>/list?type=<string>&page=<int>&pageSize=<int>

<URL>/list?name=<string>

<URL>/list?name=<string>&page=<int>&pageSize=<int>

<URL>/list?type=<string>&name=<string>

<URL>/list?type=<string>&name=<string>&page=<int>&pageSize=<int>

You can retrieve a list of all lists from the appliance where you are currently working.

You can also specify a list type, for example, string, to retrieve a list that includes only lists of this type. Similarly, you can use a list name to retrieve all lists with this name. You can combine these requests.

You can request a particular page of the feed that is returned and specify the page size.

The xml file in the feed provides a list ID for each list. This ID allows you to identify a list that you want to work with.

Sample commands

Retrieve a list of all lists on the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/list"
```

Retrieve a list of all string lists on the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/list?type=string"
```

Retrieve a list of all lists named "default" on the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/list?name=default"
```

Variable request parameters

Parameter	Type	Description
type	String	List type, which can be: • category • ip • iprange • mediatype • number • regex • string Default: None
name	String	Name of a list, for example, `default` Default: None
page	Integer	Number of a page in a feed with a list of lists Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Retrieving a list

Request format

<URL>/list/<list ID>

You can retrieve a list from the appliance where you are currently working.

When retrieving this list, you provide its ID.

The response to this request includes the retrieved list in xml format.

Sample command

Retrieve a list:

```
curl -i -b cookies.txt -X GET "$REST/list/com.scur.type.regex.4537"
```

Variable request parameter

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None

Adding a list with content

Request format

<URL>/list?type=<string>&name=<string>

You can add a list with content to the lists on the appliance where you are currently working.

When adding this list, you provide its type and name and append it in xml format as the request body. You must also specify a Content-Type header.

After adding the list, you must commit this change using a separate command.

The response to this request includes the added list in xml format as the response body.

Sample command

Add a list with content:

```
curl -i -b cookies.txt -H "Content-Type: application/xml"  -d @categoryListWithContent.xml
 -X POST "$REST/list?type=category&name=newlist"
```

Variable request parameters

Parameter	Type	Description
type	String	List type, which can be: • category • ip • iprange • mediatype • number • regex • string Default: None
name	String	Name of an added list, for example, `newlist` Default: None

Adding a list with content including type and name

Request format

<URL>/list

You can add a list with content that includes its type and name to the lists on the appliance where you are currently working.

When adding a list in this way, you do not provide a type and name, but simply append the list in xml format as the request body. You must also specify a Content-Type header.

After adding the list, you must commit this change using a separate command.

The response to this request includes the added list in xml format.

Sample command

Add a list with content that includes type and name:

```
curl -i -b cookies.txt -H "Content-Type: application/xml"
-d @listWithTypeAndNameInside.xml -X POST "$REST/list"
```

Request parameters

None

Adding an empty list

Request format

<URL>/list?type=<string>&name=<string>

You can add an empty list to the lists on the appliance where you are currently working.

When adding this list, you provide its type and name.

After adding the list, you must commit this change using a separate command.

The response to this request includes the empty list in xml format.

Sample command

Add an empty list:

```
curl -i -b cookies.txt -X POST "$REST/list?type=ip&name=emptyiplist"
```

Variable request parameters

Parameter	Type	Description
type	String	List type, which can be: • category • ip • iprange • mediatype • number • regex • string Default: None
name	String	Name of a list, for example, `emptylist` Default: None

Modifying a list

Request format

<URL>/list/<list ID>

You can modify a list on the appliance where you are currently working.

When modifying this list, you provide its ID and append the modified content in xml format as the request body. You must also specify a Content-Type header.

After modifying the list, you must commit this change using a separate command.

The response to this request includes the modified list in xml format.

Sample command

Modify a list:

```
curl -i -b cookies.txt -H "Content-Type: application/xml"
-d @modifiedContent.xml -X PUT "$REST/list/com.scur.type.regex.4537"
```

Variable request parameter

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None

Renaming a list

Request format

<URL>/list/<list ID>/rename?name=<string>

You can rename a list on the appliance where you are currently working.

When renaming this list, you provide its ID and the new name for the list.

After renaming the list, you must commit this change using a separate command.

The response to this request includes the renamed list in xml format.

Sample command

Rename a list:

```
curl -i -b cookies.txt -X POST "$REST/list/com.scur.type.regex.4537/rename?name=newname"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
name	String	Name of a renamed list, for example, `newname` Default: None

Copying a list

Request format

<URL>/list/<list ID>/copy?name=<string>

You can copy a list on the appliance where you are currently working.

When copying this list, you provide its ID and a name for the copied list.

After copying the list, you must commit this change using a separate command.

The response to this request includes the copied list in xml format.

Sample command

Copy a list:

```
curl -i -b cookies.txt -X POST "$REST/list/com.scur.type.regex.4537/copy?name=newname"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
name	String	Name of a copied list, for example, `othername` Default: None

Deleting a list

Request format

<URL>/list/<list ID>

You can delete a list on the appliance where you are currently working.

When deleting a list, you provide its ID.

Sample command

Delete a list:

```
curl -i -b cookies.txt -X DELETE "$REST/list/com.scur.type.regex.4537"
```

Variable request parameter

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None

Retrieving a list of list entries

Request formats

<URL>/list/<list ID>/entry

<URL>/list/<list ID>/entry?page=<int>&pageSize=<int>

You can retrieve a list of the entries in a list on the appliance where you are currently working.

When retrieving this list, you provide the ID of the list that contains the entries.

You can request a particular page of the feed that is returned and specify the page size.

The feed provides a list of the entries in xml format with a number for each entry to indicate its position. This number allows you to identify an entry that you want to access.

Sample command

Retrieve the entries of a list on the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/list/com.scur.type.regex.4537/entry"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
page	Integer	Number of a page in a feed with list entries Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Retrieving a list entry

Request format

<URL>/list/<list ID>/entry/<position>

You can retrieve an entry from a list on the appliance where you are currently working.

When retrieving an entry, you provide the list ID and the position of the entry in the list.

The response to this request includes the entry in xml format.

Sample command

Retrieve an entry in a list:

```
curl -i -b cookies.txt -X GET "$REST/list/com.scur.type.regex.4537/entry/2"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
position	Integer	Position of an entry in a list The position is specified as i –1. For example, for the third entry, specify 2. Default: None

Inserting a list entry

Request format

<URL>/list/<list ID>/entry/<position>/insert

You can insert an entry in a list on the appliance where you are currently working.

When inserting an entry, you provide the list ID and the position for the entry in the list.

You must also specify a Content-Type header and append the entry in xml format as the request body.

After inserting the entry, you must commit this change using a separate command.

The response to this request includes the inserted entry in xml format.

Sample command

Insert an entry in a list:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @newEntry.xml
-X POST "$REST/list/com.scur.type.regex.4537/entry/1/insert"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
position	Integer	Position for an entry in a list The position is specified as i – 1. For example, to insert an entry in third position, specify 2. Default: None

Modifying a list entry

Request format

<URL>/list/<list ID>/entry/<position>

You can modify an entry in a list where you are currently working.

When modifying this entry, you provide the list ID and the position of the entry in the list.

You must also specify a Content-Type header and append the modified content in xml format as the request body.

After modifying the entry, you must commit this change using a separate command.

The response to this request includes the modified entry in xml format.

Sample command

Modify an entry in a list:

```
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @modifiedEntry.xml
 -X PUT "$REST/list/com.scur.type.regex.4537/entry/3"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
position	Integer	Position of an entry in a list The position is specified as i – 1. For example, when modifying the third entry, specify 2. Default: None

Moving a list entry

Request format

<URL>/list/<list ID>/entry/<position>/move?newpos=<int>

You can move an entry in a list on the appliance where you are currently working.

When moving this entry, you provide the list ID and the old and new positions of the entry in the list.

After moving the entry, you must commit this change using a separate command.

The response to this request includes the entry on its new position in xml format.

Sample command

Move an entry in a list:

```
curl -i -b cookies.txt -X POST "$REST/list/com.scur.type.regex.4537/entry/4/move?newpos=1"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
position	Integer	Position of an entry in a list The position is specified as i – 1. For example, when moving an entry from second position, specify 1. Default: None
newpos	Integer	New position of an entry in a list The position is specified as i – 1. For example, to move an entry to fourth position, specify 3. Default: None

Deleting a list entry

Request format

<URL>/list/<list ID>/entry/<position>

You can delete an entry from a list on the appliance where you are currently working.

When deleting this entry, you provide the list ID and the position of the entry in the list.

Sample command

Delete an entry in a list:

```
curl -i -b cookies.txt -X DELETE "$REST/list/com.scur.type.regex.4537/entry/0"
```

Variable request parameters

Parameter	Type	Description
list ID	String	List identifier, for example, `com.scur.type.regex.4537` Default: None
position	Integer	Position of an entry in a list The position is specified as i – 1. For example, to delete the third entry, specify 2. Default: None

Sample script for working with lists

The following bash script modifies an entry in a list on the appliance you are currently working on.

Before performing this operation, the script sets a URL variable for accessing the REST interface.

Some list operations require a commit, for example, modifying a list entry. When a commit is required for an operation, this is mentioned in the description.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Modify list entry
curl -i -b cookies.txt -H "Content-Type: application/xml" -d @modifiedEntry.xml
-X PUT "$REST/list/com.scur.type.regex.4537/entry/3"
## Commit modification
curl -b cookies.txt -X POST "$REST/commit"
## Log off
curl -b cookies.txt -X POST "$REST/logout"
```

Working with rule sets

You can use the REST interface to work with rule sets and their rules on the appliance where you are currently working.

When working with rule sets, you include the `rulesets` parameter in all requests.

With rule sets, you can:

- Retrieve a list of rule sets
- Retrieve a rule set
- Export rule sets
- Export a rule set
- Import a rule set into a top-level position
- Import a rule set into a nested position
- Enable a rule set
- Disable a rule set
- Move a rule set
- Delete a rule set

Retrieving a list of rule sets

Request formats

<URL>/rulesets?topLevelOnly=<Boolean>

<URL>/rulesets?topLevelOnly=<Boolean>&page=<int>&pageSize=<int>

You can retrieve a list of all rule sets on the appliance where you are currently working.

When retrieving this list, set the value of the `topLevelOnly` parameter to `true` to include only top-level rule sets. Rule sets that are nested in a top-level rule set are not shown on the list.

You can request a particular page of the feed that is returned and specify the page size.

The feed provides a list in xml format with an ID for each rule set. This ID allows you to identify a rule set that you want to work with.

Sample commands

Retrieve a list of all rule sets on the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/rulesets?topLevelOnly=false"
```

Retrieve a list of all top-level rule sets on the appliance where you are currently working, omitting nested rule sets:

```
curl -i -b cookies.txt -X GET "$REST/rulesets?topLevelOnly=true"
```

Variable request parameters

Parameter	Type	Description
topLevelOnly	Boolean	If true, only top-level rule sets are shown on a list of rule sets. Otherwise, also nested rule sets are shown. Default: false
page	Integer	Number of a page in a feed with a list of rule sets Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Retrieving a rule set

Request formats

<URL>/rulesets/rulegroups/<rule set ID>

<URL>/rulesets/rulegroups/<rule set ID>/successor?topLevelOnly=<Boolean>

<URL>/rulesets/rulegroups/<rule set ID>/successor?topLevelOnly=<Boolean>&page=<int>&pageSize=<int>

You can retrieve a rule set on the appliance where you are currently working.

When retrieving a rule set, you append the `rulegroups` parameter in addition to the `rulesets` parameter and provide the rule set ID. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

You can retrieve a rule set with all rule sets following it on the rule sets tree by appending the `successor` parameter.

To retrieve a top-level rule set without nested rule sets, set the value of the `topLevelOnly` parameter to `true`.

You can request a particular page of the feed that is returned and also specify the page size.

The feed provides the rule set in xml format with nested rule sets and successors according to what you specified.

Sample commands

Retrieve an individual rule set on the appliance where you are currently working:

```
curl -i -b cookies.txt -X GET "$REST/rulesets/rulegroups/5234"
```

Retrieve an individual rule set on the appliance where you are currently working, including its successors and nested rule sets:

```
curl -i -b cookies.txt -X GET "$REST/rulesets/rulegroups/5234/successor?topLevelOnly=false"
```

Variable request parameters

Parameter	Type	Description
rule set ID	Integer	Rule set identifier, for example, `5234` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None
topLevelOnly	Boolean	If true, only top-level rule sets are shown on a list of rule sets Otherwise, also nested rule sets are shown. Default: false
page	Integer	Number of a page in a feed with a list of rule sets Default: 1
pageSize	Integer	Size of a page in a feed Default: –1

Exporting rule sets

Request format

<URL>/rulesets/export

You can export all rule sets that exist on the appliance where you are currently working. The exported data includes all rules, lists, settings, and properties pertaining to each of the rule sets.

When exporting rule sets, you append the `export` parameter in addition to the `rulesets` parameter.

The exported data is stored in a file with the name that you provide using the `-o` parameter.

The feed that is returned contains the exported data in xml format under the file name that you provided.

Sample command

Export all rule sets that exist on the appliance where you are currently working:

```
curl -b cookies.txt -X POST "$REST/rulesets/export" -o rulesetInMWGLibraryXMLForm.xml
```

Variable request parameters

None

Exporting a rule set

Request format

<URL>/rulesets/rulegroups/<rule set ID>/export

You can export a rule set from the appliance where you are currently working. The exported data includes all rules, lists, settings, and properties pertaining to the rule set. Nested rule sets are also included.

When exporting a rule set, you append the `export` and `rulegroups` parameters in addition to the rulesets parameter and provide the rule set ID. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

The exported data is stored in a file with the name that you provide, using the *-o* parameter.

The xml file that you receive as a feed in response to your request contains the exported data under the file name that you provided.

Sample command

Export an individual rule set that exists on the appliance where you are currently working:

```
curl -b cookies.txt -X POST "$REST/rulesets/rulegroups/5234/export"
-o rulesetInMWGLibraryXMLForm.xml
```

Variable request parameter

Parameter	Type	Description
rule set ID	Integer	Rule set identifier, for example, `5234` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None

Importing a rule set into a top-level position

Request formats

<URL>/rulesets/import

<URL>/rulesets/import?position=<int>&autoResolveBy=<string>

You can import an individual rule set into a top-level position of the rule set system on the appliance where you are currently working. The imported data includes all rules, lists, settings, and properties pertaining to the rule set.

When importing a rule set into a top-level position, you append the import `parameter` in addition to the `rulesets` parameter.

You can specify a position for the rule set and the method of resolving conflicts with existing configuration items, for example, lists and settings. If you do not specify a position or method, default values are applied.

You must also specify a Content-Type header and provide the name of the xml file that contains the rule set data, using the `-d` parameter.

The feed that is returned contains the imported rule set data in xml format.

Sample command

Import a rule set into a top-level position on the appliance where you are currently working:

```
curl -i -b cookies.txt H "Content-Type: application/xml" -d @rulesetInMWGLibraryXMLForm.xml
-X POST "$REST/rulesets/import?position=3&autoResolveBy=copy"
```

Variable request parameters

Parameter	Type	Description
position	Integer	Number of the position that an imported top-level rule set takes in the rule set system The rule set system can be viewed on the standard Web Gateway interface. Numbering begins with 0, which means if you specify 0, the rule set is imported into first position among the existing top-level rule sets. If you specify –1 or nothing at all, the rule set is imported into the last position. Default: –1
autoResolveBy	String	Method used for resolving conflicts that might arise when a rule set is imported Conflicts arise when a rule set uses configuration items, such as lists or settings, that already exist in the rule set system on your appliance. The resolution method can be: • off — No conflict resolution is performed. • copy — If items used by the imported rule set already exist under the same names in the rule set system, they are copied and renamed. The renamed items are used by the imported rule set. • refer — If items used by the imported rule set already exist under the same names in the rule set system, the rule set uses these items. • auto — If items used by the imported rule set already exist under the same name in the rule set system, the refer method is tried first, then the copy method. Usually, this solves all conflicts. Default: off If not all conflicts could be solved by applying the selected method, an error message with code number 409 is sent in response to your request. The response body then includes the conflicting data.

Importing a rule set into a nested position

Request formats

<URL>/rulesets/rulegroups/<parent rule set ID>/import

<URL>/rulesets/rulegroups/<parent rule set ID>import?position=<int>&autoResolveBy=<string>

You can import a rule set into a nested position of the rule set system on the appliance where you are currently working. The imported data includes all rules, lists, settings, and properties pertaining to the rule set.

When importing a rule set into a nested position, you append the `import` and `rulegroups` parameters in addition to the `rulesets` parameter and provide the ID of the rule set that serves as parent of the nested rule set. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

You can specify a position among the nested rule sets for the imported rule set and the method of resolving conflicts with existing configuration items, for example, lists and settings. If you do not specify a position or method, default values are applied.

You must also specify a Content-Type header and provide the name of the xml file that contains the rule set data, using the `-d` parameter.

The feed that is returned contains the imported rule set data in xml format.

Sample command

Import a rule set into a nested position on the appliance where you are currently working:

```
curl -i -b cookies.txt H "Content-Type: application/xml" -d @rulesetInMWGLibraryXMLForm.xml
-X POST "$REST/rulesets/rulegroups/4224/import?position=0&autoResolveBy=auto"
```

Variable request parameters

Parameter	Type	Description
parent rule set ID	Integer	Rule set identifier for a parent rule set, for example, `4224` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None
position	Integer	Number of the position that an imported rule set takes among the existing nested rule sets of its parent rule set. Numbering begins with 0, which means if you specify 0, the rule set is imported into first position among the existing nested rule sets. If you specify –1 or nothing at all, the rule set is imported into the last position. Default: –1
autoResolveBy	String	Method used for resolving conflicts that might arise when a rule set is imported Conflicts arise when a rule set uses configuration items, such as lists or settings, that already exist in the rule set system on your appliance. The resolution method can be: • off — No conflict resolution is performed. • copy — If items used by the imported rule set already exist under the same names in the rule set system, they are copied and renamed. The renamed items are used by the imported rule set. • refer — If items used by the imported rule set already exist under the same names in the rule set system, the rule set uses these items. • auto — If items used by the imported rule set already exist under the same name in the rule set system, the refer method is tried first, then the copy method. Usually, this solves all conflicts. Default: off If not all conflicts could be solved by applying the selected method, an error message with code number 409 is sent in response to your request. The response body then includes the conflicting data.

Enabling a rule set

Request format

<URL>/rulesets/rulegroups/<rule set ID>/enable

You can enable a rule set on the appliance where you are currently working.

If this rule set is nested within a parent rule set, make sure this rule set is also enabled before enabling the nested rule set.

When enabling a rule set, you append the `enable` and `rulegroups` parameters in addition to the `rulesets` parameter and provide the rule set ID. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

You must also specify a Content-Type header.

After enabling the rule set, you must commit this change using a separate command.

Sample command

Enable a rule set on the appliance where you are currently working:

```
curl -i -b cookies.txt H "Content-Type: application/xml"
-X POST "$REST/rulesets/rulegroups/1929/enable"
```

Variable request parameters

Parameter	Type	Description
rule set ID	Integer	Rule set identifier, for example, `5234` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None

Disabling a rule set

Request format

<URL>/rulesets/rulegroups/<rule set ID>/disable

You can disable a rule set on the appliance where you are currently working.

When disabling a rule set, you append the `disable` and `rulegroups` parameters in addition to the `rulesets` parameter and provide the rule set ID. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

You must also specify a Content-Type header.

After disabling the rule set, you must commit this change using a separate command.

Sample command

Disable a rule set on the appliance where you are currently working:

```
curl -i -b cookies.txt H "Content-Type: application/xml"
-X POST "$REST/rulesets/rulegroups/1929/disable"
```

Variable request parameters

Parameter	Type	Description
rule set ID	Integer	Rule set identifier, for example, `5234` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None

Moving a rule set

Request formats

<URL>/rulesets/rulegroups/<rule set ID>/move?position=<int>

<URL>/rulesets/rulegroups/<rule set ID>/move?parentId=<int>&position=<int>

You can move a rule set to a different position within the rule set system on the appliance where you are currently working. You can move it to a top-level or a nested position.

When moving a rule set to a top-level position, you append the `move` and `rulegroups` parameters in addition to the `rulesets` parameter and provide the rule set ID. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

You must also provide a position number to specify the top-level position, for example, the first position among all existing top-level rule sets.

When moving a rule set to a nested position, you append the `move` and `rulegroups` parameters in addition to the `rulesets` parameter and provide the rule set IDs of the moved rule set and the rule set that serves as its parent.

You must also provide a position number to specify the position of the moved rule set among the existing nested rule sets of the parent rule set.

After moving the rule set, you must commit this change using a separate command.

Sample commands

Move a rule set to a top-level position on the appliance where you are currently working:

```
curl -i -b cookies.txt -X POST "$REST/rulesets/rulegroups/4224/move?position=2"
```

Move a rule set to a nested position on the appliance where you are currently working:

```
curl -i -b cookies.txt -X POST "$REST/rulesets/rulegroups/6326/move?parentId=2159&position=4"
```

Variable request parameters

Parameter	Type	Description
rule set ID	Integer	Rule set identifier, for example, `5234` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None
parentId	Integer	Rule set identifier for a parent rule set, for example, `4224` Default: None
position	Integer	Number of the position that an moved rule set takes among the existing top-level rule sets or nested rule sets of its parent rule set. Numbering begins with 0, which means if you specify 0, the rule set is moved into first position among the existing top-level or nested rule sets. If you specify –1 or nothing at all, the rule set is imported into the last position. Default: –1

Deleting a rule set

Request format

<URL>/rulesets/rulegroups/<rule set ID>/delete

You can delete a rule set on the appliance where you are currently working.

When deleting a rule set, you append the `delete` and `rulegroups` parameters in addition to the `rulesets` parameter and provide the rule set ID. You can look up rule set IDs in the feed that is returned when you retrieve a list of all rule sets.

After deleting the rule set, you must commit this change using a separate command.

Sample command

Delete a rule set on the appliance where you are currently working:

```
curl -i -b cookies.txt H "Content-Type: application/xml"
-X DELETE "$REST/rulesets/rulegroups/2037/delete"
```

Variable request parameters

Parameter	Type	Description
rule set ID	Integer	Rule set identifier, for example, `5234` Rule set IDs are included in the feed that is returned when you retrieve a list of all rule sets. Default: None

Sample script for working with rule sets

The following bash script deletes a rule set on the appliance you are currently working on.

Before performing this operation, the script sets a URL variable for accessing the REST interface.

```
#!/bin/bash
## Set URL variable for accessing REST interface
REST=http://localhost:4711/Konfigurator/REST
## Log on and authenticate
curl -i -c cookies.txt -X POST "$REST/login?userName=myUserName&pass=myPassword"
## Delete rule set
curl -i -b cookies.txt -X DELETE "$REST/rulesets/rulegroups/com.scur.type.regex.4537/entry/3"
## Commit deletion
curl -b cookies.txt -X POST "$REST/commit"
## Log off
curl -b cookies.txt -X POST "$REST/logout"
```