Cisco ASA REST API Quick Start Guide

First Published: 2014-12-24

Last Modified: 2020-09-16

Cisco ASA REST API Quick Start Guide



Note The REST API is not supported on newer hardware models and is no longer being developed. We recommend that you instead use the ASA HTTP interface for automation. See Cisco Secure Firewall ASA HTTP Interface for Automation.

Overview

Several options are available for configuring and managing individual Cisco ASAs:

- Command Line Interface (CLI) – you send control commands directly to the ASA via a connected console.
- Adaptive Security Device Manager (ASDM) – an "on-box" management application with a graphical user interface that you can use to configure, manage and monitor an ASA.
- Cisco Security Manager – while intended for medium to large networks of many security devices, this graphical application can be used to configure, manage and monitor individual ASAs.

With the release of Cisco's ASA REST API, you now have another light-weight, easy-to-use option. This is an application programming interface (API), based on "RESTful" principles, which you can quickly download and enable on any ASA on which the API is running.

After installing a REST client in your browser, you can contact the specific ASA's REST agent and use standard HTTP methods to access current configuration information, and issue additional configuration parameters.



Caution When the REST API is enabled on an ASA, connections by other security management protocols are not blocked. This means others using CLI, ASDM, or Security Manager could be altering the ASA configuration while you are doing the same.

ASA REST API Requests and Responses

The ASA REST API gives you programmatic access to managing individual ASAs through a Representational State Transfer (REST) API. The API allows external clients to perform CRUD (Create, Read, Update, Delete) operations on ASA resources; it is based on the HTTPS protocol and REST methodology.

All API requests are sent over HTTPS to the ASA, and a response is returned.

This section provides an overview of how requests are structured, and the expected responses,

Request Structure

Available request methods are:

- GET – Retrieves data from the specified object.
- PUT – Adds the supplied information to the specified object; returns a 404 Resource Not Found error if the object does not exist.
- POST – Creates the object with the supplied information.
- DELETE – Deletes the specified object.
- PATCH – Applies partial modifications to the specified object.

Response Structure

Each request produces an HTTPS response from the ASA with the standard headers, response content, and status code.

The response structure can be:

- LOCATION – Newly created resource ID; for POST only—holds the new resource ID (as a URI representation).
- CONTENT-TYPE – Media type describing the response message body; describes the representation and syntax of the response message body.

Each response includes an HTTP status or error code. Available codes fall into these categories:

- `20x` - A two-hundred series code indicates successful operation, including:
  - `200 OK` – Standard response for successful requests.
  - `201 Created` – Request completed; new resource created.
  - `202 Accepted`– Request accepted, but processing not complete.
  - `204 No Content`– Server successfully processed request; no content is being returned.
- `4xx` - A four-hundred series code indicates a client-side error, including:
  - `400 Bad Request` – Invalid query parameters, including unrecognized parameters, missing parameters, or invalid values.
  - `404 Not Found` – The provided URL does not match an existing resource. For example, an HTTP DELETE may fail because the resource is unavailable.
  - `405 Method not Allowed` – An HTTP request was presented that is not allowed on the resource; for example, a POST on a read-only resource.
- `5xx` - A five-hundred series code indicates a server-side error.

In the case of an error, in addition to the error code, the return response may include an error object containing more details about the error. The JSON error/warning response schema is as follows:

```
[
{ "code" : "string",
"details": "string",
"context": attribute name,
"level" : <Error/Warning/Info>
},
...
]
```

where the object properties are:

Property	Type	Description
`messages`	List of Dictionaries	List of error or warning messages
`code`	String	Error/Warning/Info code
`details`	String	Detailed message corresponding to Error/Warning/Info

Note Changes to the ASA configuration made by REST API calls are not persisted to the startup configuration; that is, changes are assigned only to the running configuration. To save changes to the startup configuration, you can POST `a writemem` API request; for more information, follow the "Write Memory API" entry in the About the ASA REST API table of contents.

Install and Configure the ASA REST API Agent and Client

The REST API Agent is published individually with other ASA images on cisco.com. For physical ASAs, the REST API package must be downloaded to the device's flash and installed using the "rest-api image" command. The REST API Agent is then enabled using the "rest-api agent" command.

With a virtual ASA (ASAv), the REST API image must be downloaded to the "boot:" partition. You must then issue the "rest-api image" command, followed by the "rest-api agent" command, to access and enable the REST API Agent.

For information about REST API software and hardware requirements and compatibility, see the Cisco ASA Compatibility matrix.

You can download the appropriate REST API package for your ASA or ASAv from software.cisco.com/download/home. Locate the specific Adaptive Security Appliances (ASA) model and then choose Adaptive Security Appliance REST API Plugin.

Note The REST API Agent is a Java-based application. The Java Runtime Environment (JRE) is bundled in the REST API Agent package.

Usage Guidelines

Important You must include the header `User-Agent: REST API Agent` in all API calls and existing scripts. Use `-H 'User-Agent: REST API Agent'` for the CURL command.

In multi-context mode, the REST API Agent commands are available only in the System context.

Maximum Supported Configuration Size

The ASA Rest API is an "on-board" application running inside the physical ASA, and as such has a limitation on the memory allocated to it. Maximum supported running configuration size has increased over the release cycle to approximately 2 MB on recent platforms such as the 5555 and 5585.

The ASA Rest API also has memory constraints on the virtual ASA platforms. Total memory on the ASAv5 can be 1.5 GB, while on the ASAv10 it is 2 GB. The Rest API limits are 450 KB and 500 KB for the ASAv5 and ASAv10, respectively.

Therefore, be aware that large running configurations can produce exceptions in various memory-intensive situations such as a large number of concurrent requests, or large request volumes. In these situations, Rest API GET/PUT/POST calls may begin failing with 500 - Internal Server Error messages, and the Rest API Agent will restart automatically each time.

The workarounds to this situation are either move to higher-memory ASA/FPR or ASAV platforms, or reduce the size of the running configuration.

Maximum Supported Configuration Size

The ASA Rest API is an "on-board" application running inside the physical ASA, and as such has a limitation on the memory allocated to it. Maximum supported running configuration size has increased over the release cycle to approximately 2 MB on recent platforms such as the 5555 and 5585.

The ASA Rest API also has memory constraints on the virtual ASA platforms. Total memory on the ASAv5 can be 1.5 GB, while on the ASAv10 it is 2 GB. The Rest API limits are 450 KB and 500 KB for the ASAv5 and ASAv10, respectively.

Therefore, be aware that large running configurations can produce exceptions in various memory-intensive situations such as a large number of concurrent requests, or large request volumes. In these situations, Rest API GET/PUT/POST calls may begin failing with 500 - Internal Server Error messages, and the Rest API Agent will restart automatically each time.

The workarounds to this situation are either move to higher-memory ASA/FPR or ASAV platforms, or reduce the size of the running configuration.

Download and Install the REST API Agent

Using the CLI, follow these steps to download and install the ASA REST API agent on a specific ASA:

Procedure

Step 1 On the desired ASA, issue the copy `<package> disk0:` command to download the current ASA REST API package from cisco.com to the ASA's flash memory. For example:

```
copy tftp://10.7.0.80/asa-restapi-111-lfbff-k8.SPA disk0:
```

Step 2 Issue the `rest-api image disk0:/<package>` command to verify and install the package.

For example:

```
rest-api image disk0:/asa-restapi-111-lfbff-k8.SPA
```

The installer will perform compatibility and validation checks, and then install the package. The ASA will not reboot.

Enable the REST API Agent

Follow these steps to enable the ASA REST API Agent on a specific ASA:

Procedure

Step 1 Ensure the correct software image is installed on the ASA.

Consult the REST API section of the ASA Compatibility Matrix https://www.cisco.com/c/en/us/td/docs/security/asa/compatibility/asamatrx.html#pgfId-131643 to determine which ASA image is required.

Step 2 Using the CLI, ensure the HTTP server is enabled on the ASA, and that API clients can connect to the management interface. For example:

```
http server enable
http 0.0.0.0 0.0.0.0 <management interface nameif>
```

Step 3 Using the CLI, define HTTP authentication for the API connections. For example:

```
aaa authentication http console LOCAL
```

Step 4 Using the CLI, create a static route on the ASA for API traffic. For example:

```
route <management interface nameif> 0.0.0.0 0.0.0.0 <gwip> 1
```

Step 5 Using the CLI, enable the ASA REST API Agent on the ASA. For example:

```
rest-api agent
```

REST API Authentication

There are two ways to authenticate: Basic HTTP authentication, which passes a user name and password in every request, or Token-based authentication with secure HTTPS transport, which passes a previously created token with each request. Either way, authentication will be performed for every request. See the section, "Token_Authentication_API" in the *About the ASA REST API v7.14(x)* guide for additional information about Token-based authentication.



Note Use of Certificate Authority (CA)-issued certificates is recommended on ASA, so REST API clients can validate the ASA server certificates when establishing SSL connections.

Command Authorization

If command authorization is configured to use an external AAA server (for example, `aaa authorization command <TACACS+_server>`), then a user named **enable_1** must exist on that server with full command privileges.

If command authorization is configured to use the ASA's LOCAL database (`aaa authorization command LOCAL`), then all REST API users must be registered in the LOCAL database with privilege levels that are appropriate for their roles:

- Privilege level 3 or greater is required to invoke monitoring requests.
- Privilege level 5 or greater is required for invoking GET requests.
- Privilege level 15 is necessary for invoking PUT/POST/DELETE operations.

Configure Your REST API Client

Follow these steps to install and configure a REST API client on your local-host browser:

Procedure

Step 1 Acquire and install a REST API client for your browser.

For Chrome, install the REST client from Google. For Firefox, install the RESTClient add-on. Internet Explorer is not supported.

Step 2 Initiate the following request using your browser:

`https:<asa management ip address>/api/objects/networkobjects`

If you receive a non-error response, you have reached the REST API agent functioning on the ASA.

If you are having issues with the agent request, you can enable display of debugging information on the CLI console, as described in Enabling REST API Debugging on the ASA.

Step 3 Optionally, you can test your connection to the ASA by performing a POST operation.

For example:

Provide basic authorization credentials *(<username><password>)*, or an authentication token (see Token Authentication for additional information).

Target request address: `https://<asa management ipaddress>/api/objects/networkobjects`

Body content type: `application/json`

Raw body of the operation:

```
{
"kind": "object#NetworkObj",
"name": "TestNetworkRangeObj",
"host": {
"kind": "IPv4Network",
"value": "12.12.12.0/24"
}
}
```

You can now use the ASA REST API to configure and monitor the ASA. Refer the API documentation for call descriptions and examples.

About Fully Restoring a Back-up Configuration

Restoring a full back-up configuration on the ASA using the REST API will reload the ASA. To avoid this, use the following command to restore a back-up configuration:

```
{
"commands":["copy /noconfirm disk0:/<filename> running-config"]
}
```

Where `<filename>` is backup.cfg or whatever name you used when backing up the configuration.

The Documentation Console and Exporting API Scripts

You also can use the REST API on-line documentation console (referred to as the "Doc UI"), available at *host:port/doc/* as a "sandbox" for learning about and trying the API calls directly on the ASA.

Further, you can use the **Export Operation** button in the Doc UI to save the displayed method example as a JavaScript, Python, or Perl script file to your local host. You can then apply this script to your ASA, and edit it for application on other ASAs and other network devices. This meant primarily as an educational and bootstrapping tool.

JavaScript

Using a JavaScript file requires installation of `node.js`, which can be found at `http://nodejs.org/`. Using `node.js`, you can execute a JavaScript file, typically written for a browser, like a command-line script. Simply follow the installation instructions, and then run your script with `node script.js`.

Python

The Python scripts require you to install Python, available from `https://www.python.org/`. Once you've installed Python, you can run your script with `python script.py username password`.

Perl

Using the Perl scripts requires some additional set-up—you need five components: Perl itself, and four Perl libraries:

- Perl package, found at `http://www.perl.org/`
- Bundle::CPAN, found at `http://search.cpan.org/~andk/Bundle-CPAN-1.861/CPAN.pm`
- REST::Client, found at

  `http://search.cpan.org/%7Emcrawfor/REST-Client-88/lib/REST/Client.pm`
- MIME::Base64, found at `http://perldoc.perl.org/MIME/Base64.html`
- JSON, found at `http://search.cpan.org/~makamaka/JSON-2.90/lib/JSON.pm`

Here is an example of bootstrapping Perl on a Macintosh:

```
$ sudo perl -MCPAN e shell
cpan> install Bundle::CPAN
cpan> install REST:: Client
cpan> install MIME::Base64
cpan> install JSON
```

After installing the dependencies, you can run your script using `perl script.pl username password`.

Enabling REST API Debugging on the ASA

If you are having problems configuring or connecting to the REST API on the ASA, you can use the following CLI command to enable display of debugging messages on your console. Use the no form of the command to disable the debug messages.

debug rest-api [**agent** | **cli** | **client** | **daemon** | **process** | **oken-auth**] [**error** | **event**]

no debug rest-api

Syntax Description

agent	(Optional) Enable REST API Agent debugging information.
cli	(Optional) Enable debugging messages for REST API CLI Daemon-to-Agent communications.
client	(Optional) Enable debugging information for Message routing between the REST API Client and the REST API Agent.
daemon	(Optional) Enable debugging messages for REST API Daemon-to-Agent communications.
process	(Optional) Enable REST API Agent process start/stop debugging information.
token-auth	(Optional) REST API token authentication debugging information.
error	(Optional) Use this keyword to limit debug messages to only errors logged by the API.
event	(Optional) Use this keyword to limit debug messages to only events logged by the API.

Usage Guidelines

If you do not provide a specific component keyword (that is, if you simply issue the command **debug rest-api**), debug messages are displayed for all component types. If you do not provide either the **event** or **error** keyword, both event and error messages are displayed for the specified component. For example, **debug rest-api daemon event** will show only event debug messages for API Daemon-to-Agent communications.

Related Commands

Command	Description
debug http	Use this command to view detailed information about HTTP traffic.

ASA REST API-related Syslog Messages

The ASA REST API-related system-log messages are described in this section.

Related Documentation

Use the following link to find more information about the ASA, and its configuration and management:

- *Navigating the Cisco ASA Series Documentation:* http://www.cisco.com/go/asadocsxm-replace_text%20List%20Item

Use the following link to view a list of ASA features not supported on the ASAv:

- http://www.cisco.com/c/en/us/td/docs/security/asa/asa92/configuration/general/asa-general-cli/introasav.html#pgfId-1156883

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2020 Cisco Systems, Inc. All rights reserved.