Cisco FXOS REST API Getting Started Guide

Americas Headquarters

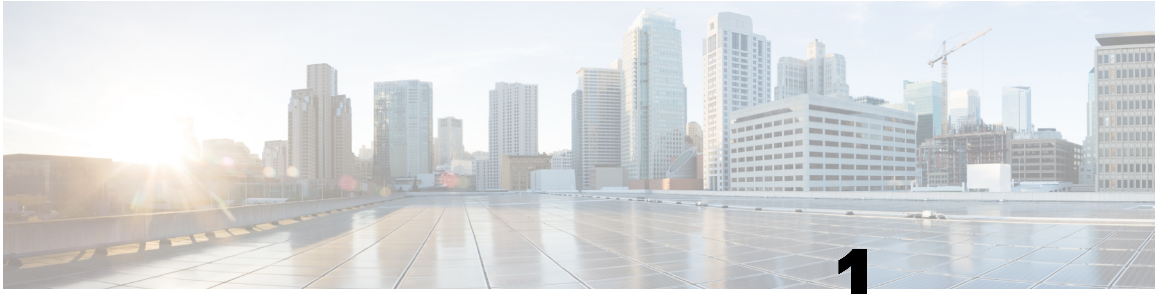
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

CONTENTS



CHAPTER 1

Introduction to the FXOS REST API

You can use the FXOS REpresentational State Transfer (REST) Application Programming Interface (API), over HTTPS, to interact with FXOS device services through a client program. The REST API uses JavaScript Object Notation (JSON) format to represent objects.

The “FXOS API” encompasses both Platform and Firepower Chassis Services APIs, including both Configuration and Monitoring APIs. These APIs are solely for the Platform services provided by the Firepower eXtensible OS (FXOS).

The FXOS API includes an API Explorer that describes the resources and JSON objects available for your programmatic use. The Explorer provides detailed information about the attribute-value pairs in each object, and you can experiment with the various HTTP methods to ensure you understand the use of each resource. The API Explorer also provides examples of the URLs required for each resource.

You can also find reference information and examples online at https://developer.cisco.com/docs/cisco-firepower-chassis-manager-rest-api-reference/#!cisco-firepower-chassis-manager-rest-api-reference/introduction.

Accessing the FXOS API Explorer

The FXOS API Explorer describes the resources and JSON objects available for your programmatic use. The Explorer provides detailed information about the attribute-value pairs in each object, and you can experiment with the various HTTP methods to ensure you understand the coding required to use each resource.

Procedure

Step 1 Using a browser, open the home page of the system, for example, `https://fcm.example.com`.

Step 2 Log into Firepower Chassis Manager.

Step 3 Open the FXOS API Explorer by performing either of the following:

a) Edit the URL to point to `/api/api-explorer/index.html`.

For example, `https://fcm.example.com/api/api-explorer/index.html`.

b) Open the FCM **Help** menu and choose **API Explorer**.

The FXOS API Explorer opens in a separate window.

Audience for This API Guide

This guide assumes you have a general knowledge of programming and a specific understanding of REST APIs and JSON. If you are new to these technologies, please refer to some general REST API reference materials.

CHAPTER 2

Getting Started with the FXOS REST API

You can access the FXOS REST API using any REST API client. Typically REST API clients are available as browser plug-ins, but any REST API client is allowed.

Base URL

All users must access the FXOS API using HTTPS via the device's configured port. In general, the base URL for all API requests will be as follows:

```
https://<Firepower_Chassis_fqdn>[:management_port]/api/
```

Cisco recommends that the Firepower chassis fully qualified domain name (`chassis_fqdn`) be used instead of the IP address in order for the REST client to be able to authenticate the device being managed; however, the IP address can be used. The Firepower chassis will authenticate the REST client using the provided credentials. See Client Authentication for additional details.

Client Authentication

The FXOS REST API uses a session-based method of authentication, requiring an HTTP `TOKEN` header in order to authenticate each request from an API client.

All API requests except `/api/login` must include this HTTP `TOKEN` header, with the `TOKEN` value being the actual token obtained in the `/api/login` response. If the `TOKEN` is not specified in the header, an Authentication Required error code will be returned by the REST API.

To obtain an authentication token, an API client must execute a `POST /api/login` call with the following HTTP headers: `username:` <*name_of_user*> `password:` <*user_password*>.

Note These user credentials must already exist on the chassis, or an Authentication Failed error will be returned. Also, authentication will fail if the user password is URL-encoded.

A successful `/api/login` call will return an HTTP 200 code with this payload:

```
{
  "refreshPeriod": "string",
  "token": "string"
}
```

For example:

```
{
  "refreshPeriod": "0",
  "token": "a205739a4d56359730c08ee7bda559cd6e4dd36ad65bb10a17f71d4cea356028"
}
```

You must extract this token and include it in the HTTP header when submitting all other API method requests, and the header field name for the token must be `TOKEN`.

On authentication failure, the API will return an HTTP 400 code with this payload: { "error-code": "551", "error-description": "Authentication failed", "error-nr": "unidentified-fail" }

On authorization failure, the API will return an HTTP 400 code with the following payload when the token is invalid: { "error-code": "552", "error-description": "Authorization required", "error-nr": "service-unavailable" }

To log out, the client must submit a `POST /api/logout` request with this HTTP header: `"TOKEN"= "<authentication_token>"`

An HTTP 200 code is returned when a session was found and that session was terminated.

An HTTP 555 code is returned with the following payload when a valid session is not found for the specified token: { "error-code": "555", "error-description": "Authorization required", "error-nr": "service-unavailable" }

Supported HTTP Methods

You can use the following HTTP methods:

- **GET** – Retrieves the specified object. The query is submitted in the URL and the output is contained in the response body.
- **POST** – Creates an object. The path is submitted in the URL and the output is contained in the response body.
- **PUT** – Updates an object completely. Read-only properties are ignored and writable properties are set to their default values if not specified in the payload. Child objects are deleted if they are not specified in the request payload (except for automatically created objects).
- **PATCH** – Partially updates the object specified in the URL. Only the included properties of the specified object will be updated. Other existing properties will retain their previous values.
- **DELETE** – Deletes the object specified in the URL.

Accept Header and Content Type

All REST API clients must send an HTTP `Accept Header` that is set to `application/json`. Otherwise a `406 Not Acceptable` error will be returned.

When uploading JSON objects (POST for creating objects, or PUT for updating objects), the `Content-type` HTTP header must be set to `application/json`. Otherwise, a `415 Unsupported Content Type` error will be returned by the API. When uploading Firepower Chassis infrastructure bundles or CSP binaries, the REST API client must set the `Content-type` HTTP header to `multipart/binary`.

HTTP Error Codes

When executing REST API methods, possible HTTP error codes are:

- **200** – Success with output. Method succeeded and equivalent JSON objects are returned.
- **204** – Success with empty output. Method is successful but the content is empty.
- **400** – Request failed. See the error message for more information.
- **403** – Forbidden. Permission Denied. The current user is not allowed to perform the operation.
- **404** – Object not found. The requested object was not found.
- **405** – Method not supported. The requested method is not supported by the API.
- **406** – Not acceptable. Returned when the client requests anything but `application/json`.
- **415** – Unsupported content-type. Returned when the client sends a content-type that is anything but `application/json`.
- **500** – Internal Error. The API encountered an error in serving the request.