# Cisco Evolved Programmable Network Manager RESTful API

## Cisco Evolved Programmable Network Manager SDK

The Cisco Evolved Programmable Network Manager SDK is a collection of technologies that enables you to extend the capabilities of Cisco Evolved Programmable Network Manager, access data, and invoke the automation operations from any application. The Cisco Evolved Programmable Network Manager SDK includes the RESTful APIs and Open Automation. It is possible to use the RESTful API with scripting languages such as "bash with wget and cURL utilities", "Python", "Ruby", Java and so on.

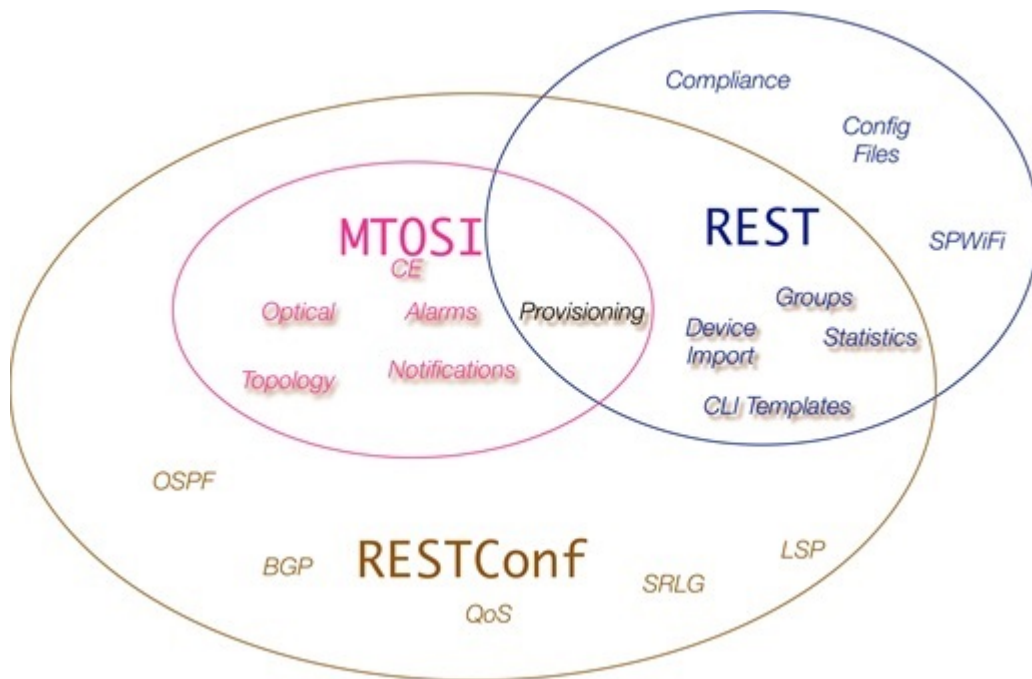With Cisco Evolved Programmable Network Manager SDK technologies, you can:

- Access Cisco Evolved Programmable Network Manager programmatically—Use the Cisco Evolved Programmable Network Manager RESTful API to invoke workflows and obtain reports.

- Customize Cisco Evolved Programmable Network Manager —Create custom workflow tasks, Customize Cisco Evolved Programmable Network Manager by deploying your own jar files and script libraries in script modules. Use custom tasks from script bundles.

## Cisco Evolved Programmable Network Manager APIs

Cisco Evolved Programmable Network Manager provides easy to use and comprehensive APIs that allow integration with any standards-based OSS system north bound. These APIs extend its core functionalities using three of the most commonly used standards - MTOSI and RESTConf, and RESTful APIs.

To provide rapid development design for automation, it is important for developers to develop APIs consistently thus paving way for a smoothest and easy to use design guidelines. Consistency allows teams to leverage common code, patterns, documentation and design decisions. In Cisco Evolved Programmable Network Manager, RESTful APIs are easily made available within the application with simple and extensive examples. The following diagram is an example that illustrates the general behavior found in Cisco Evolved Programmable Network Manager across *all* its APIs.

*Figure 1: EPNM RESTful API General Behavior*



This chapter is intended for the following technical professionals interested in using the Cisco Evolved Programmable Network Manager Development Kit (SDK) and related technologies. Such users might be:

- System administrators and REST API developers who use Cisco Evolved Programmable Network Manager and want to extend their ability to automation resources using it.

- Anyone else who wants to compare Cisco Evolved Programmable Network Manager SDK or API technologies and determine which would be most appropriate for their application

Cisco Evolved Programmable Network Manager provides three northbound API interface types, MTOSI, REST and RESTConf. The functionality provided by these three APIs are fairly similar but not all APIs provide the same functions.

The MTOSI API is the most different of the APIs in terms of behavior. It provides a SOAP interface over HTTPS that provides basic provisioning of optical and carrier ethernet functionality. Use the MTOSI interface if you need to provision optical services such as DWDM or OTN, and cannot use the RESTConf API.

The REST API provides system information and most statistics and assurance information besides other functions that handle configuration files, group management and device import capabilities.

The RESTConf API, like the above REST API, adheres to a RESTful interface. It provides all provisioning for carrier ethernet, L2/L3 VPNs, circuit emulation, OTN and DWDM technologies as well as core routing and switching.

For more information see the individual Cisco EPNM integration guides available on cisco.com.

# When to Use Cisco Evolved Programmable Network Manager RESTful API

Cisco Evolved Programmable Network Manager RESTful API is a language-independent interface that can be used by any program or script capable of making HTTP requests. Use the REST API when you want to invoke operations on Cisco Evolved Programmable Network Manager from a separate program or process.

Applications can use the RESTful API to do the following:

- Retrieve Cisco Evolved Programmable Network Manager reports on physical and virtual devices, networks, appliances, groups and users, policies, and other monitored entities within your Cisco Evolved Programmable Network Manager domains.

- Invoke additional operations specific to Cisco Evolved Programmable Network Manager.

# How to Use Cisco Evolved Programmable Network Manager RESTful API

Because a RESTful API client interacts with Cisco Evolved Programmable Network Manager using standard HTTP requests and responses, the RESTful API responses are compatible with any web browser. Many programming languages have libraries that are devoted to creating and sending HTTP requests and handling HTTP responses.

Most of the REST API calls send and return data in the request or the response, respectively. These data payloads may be formatted in one of two ways depending on the RESTful API call. Some of the RESTful API calls use a JavaScript Object Notation (JSON) payload, while others use an XML payload. You probably have to use both for any reasonably complex application.

A JSON-based RESTful API call is a plain HTTP request and response with a JSON payload. JSON is a lightweight text-based open standard that is designed for human-readable data interchange. JSON represents simple data structures and associative arrays. Your application directly invokes the JSON-based API without using any specialized RESTful API libraries, and parses JSON data using any means native to your application.

All requests to the Cisco Evolved Programmable Network Manager API require user authentication. For more information about Authentication, see Authentication and Authorization.

Authorization in the RESTful API is enforced by requiring that only registered users of Cisco Evolved Programmable Network Manager are able to make API requests. For Cisco Evolved Programmable Network Manager, access to the API is controlled by three user groups such as NBI Read and NBI Write. Each of these groups controls access to a different set of APIs. You can assign a user to multiple groups if you wish. You can check the documentation page of an API resource to determine which user group is required to access it. When a user is created and registered on Cisco Evolved Programmable Network Manager, the user is assigned a unique RESTful API access key.

To obtain the RESTConf topological link retrieval resources, you can use RESTful topological link resources to get these values and perform the required correlation.

To access Cisco Evolved Programmable Network Manager RESTful API documentation:

- Launch Cisco Evolved Programmable Network Manager, and then in the top-right corner, click ⚙, the window settings menu opens.

Logged In As root

    Log out

    Change Password

    Set Current Page As Home

    My Preferences

    Support Cases

Virtual Domain:**ROOT-DOMAIN**

Help

    Getting Started

    Online Help

    API Help

    Supported Devices

    MSE Installation Guide

    Documentation Home Page

Feedback

    I wish this page would...

About Cisco EPN Manager

- Choose **Help** > **API Help** > **REST API** to learn about system requirements, how to set up a development environment, how to install libraries, usage of RESTful APIs, the list of every Cisco Evolved Programmable Network Manager REST API function, REST API Resources, different use case examples, queries, and so on.

# RESTConf API-An Overview

In Cisco EPN Manager, the implementation conforms to the RESTConf/Yang specification information model and operational APIs protocols. Where required, the information model and operations APIs are extended in a standard way to support Cisco's vendor extensions to the RESTConf/Yang interface. These extensions are added as a set of xml and yang schema definitions for the information model extensions.

RESTCONF—Uses structured data (XML or JSON) and YANG to provide a REST-like APIs, enabling you to programmatically access different network devices. RESTCONF APIs use HTTPs methods.

YANG—A data modelling language that is used to model configuration and operational features . YANG determines the scope and the kind of functions that can be performed by NETCONF and RESTCONF APIs.

Following are the essential structure of RESTConf API:

- HTTP Headers: HTTP headers are used to describe the content sent or requested within an HTTP request. HTTP headers include:

    - Content-Type: At server side, an incoming request may have an entity attached to it. To determine its type, server uses the HTTP request header Content-Type.

    - Accept: Similarly, to determine what type of representation is desired at client side, HTTP header ACCEPT is used. Generally, if no Accept header is present in the request, the server can send pre-configured default representation type.

- **HTTP Methods**: The following methods are used to call an API:

    - Get-The GET method is sent by the client to retrieve data and metadata for a resource.

    - Post-It is used to create a new entity, but it can also be used to update an entity. The POST method is sent by the client to create a data resource or invoke an operation resource.

    - Put-A PUT request is idempotent. The PUT method is sent by the client to create or replace the target data resource.

    - Delete-Request that a resource be removed. The DELETE method is used to delete the target resource.

- **Messages**: The RESTCONF protocol uses HTTP messages. A single HTTP message corresponds to a single protocol method.

- Media Types: XML and JSON.

- **Query Parameters:**

    - Content

    - Depth

    - Fields

    - Filter

    - Insert

    - Point

    - Start-time

    - Stop-time

    - With-defaults

### Standard usage - Get Method

The "GET All RESTConf API calls" returns the existing endpoint schema sections.

### GET /restconf/data/ietf-yang-library:modules-state

Retrieves all RESTConf API endpoints

The schema URLs provide the details for many of the enumeration variables as well as the structural components of the module.

### Example for schema retrieval:

**GET /restconf/schema/v1/cisco-resource-optical**

Returns the YANG schema for the module as plain text.

Several URL parameters are often available in RESTConf API calls. These are not implemented in all calls.

- .maxCount

- .startIndex

- .depth

```
GET /restconf/data/v1/module:resource
The general format for a RESTConf GET call.
```

*Table 1: Query Parameters*

| Query Parameter | Description |
| --- | --- |
| .depth (integer) | The number of detail levels to retrieve. |
| .maxCount (integer) | The limit of the number of rows retrieved. |
| .startIndex (integer) | The initial row to retrieve. |

```
GET /restconf/data/v1/module:resource HTTP/1.1
```

**RESTful API Utility**

- Fully Distinguishable Name: Inventory objects in this interface has attributes representing FDN (Fully Distinguished Name). These attributes are used as identifiers of the object or as a reference to the object in query parameters or in the returns data wherever a reference to the object is needed. This FDN is a formatted string that consists of a set of type/value pairs with the following syntax, Sequence of <type>=<value> pairs separated by "!" where:

  - <type> is a constant value defined in the data model to represent the inventory object in the hierarchy, e.g. MD,ND,EQ,PTP,FTP, CTP, TL,VC,CFS, etc.

  - <value> is any text or sequence of <attrName>=<attrValue> pair separated by ";" that represents the attribute/value pairs of the inventory object constitutes a unique value within the local scope of the object represented by the type.

For more information, refer to the Cisco EPNM RESTConf guide.

# RESTConf API Functional Areas

- Alarms—The alarms module provides mechanisms to retrieve and acknowledge alarms and events. The preferred method of managing alarms is to listen via the Notification API where a service can collect events of differing types or criticality and then handle them via this API.

- Performance Test—The Service Performance Test can be conducted standalone as well as part of service provisioning. Restconf NBI supports standalone performance test execution.

- Quality of Service (QoS)— QoS is a set of capabilities that allow the delivery of differentiated services for network traffic. Using Cisco EPN Manager you can configure QoS on Carrier Ethernet interfaces.

- OAM—The OAM tests in the EPNM RESTConf API fall into two general categories. The service and the network resource OAM configurations. For Y.1731, Y.1564 and BERT tests, the service-oam-config endpoint is used. For OTDR, the network-resource-oam-config endpoint is used. EPNM RESTConf OAM calls provide a POST command to initiate the test. That request yields a test ID and a test request ID. One usually then uses the test ID in a subsequent call to retrieve the results. There are URLs for most specific test types.

- Service Profiles—Service Profiles contain pre-defined provisioning request (order data) that can be used in provisioning each circuit/VC type. In NBI Provisioning request, a service profile reference can be used to get provisioning request data to be used in the provisioning. If the provisioning data is provided in the request along with the service profile reference, but the user provided data and the data stored in the service profile gets merged with user provided data overriding the profile data before the request is sent to execute provisioning. Service profile can be created using EPNM Service Profile wizard GUI.

- Customer Facing Services (CFS)—The CFS represents the customer facing data for a circuit/VC. The CFS is derived from discovered RFS and represents the endpoints of the circuit/VC in the network. During CFS discovery, the system creates CFS objects for the discovered RFS objects.

- Resource Facing Services (RFS)—The RFS represents the relations between resources on different devices. During RFS discovery, the system creates device-level objects and network-level objects. Device-level RFS objects represent the circuit/VC configuration parts of the device-level configuration. Network-level RFS objects aggregate device or other network-level objects to represent network-level entities.

# Authentication and Authorization

All requests to the Cisco Evolved Programmable Network Manager API require user authentication. If no authentication details are provided in the request, the request is redirected to the login page. Authentication details may be passed through the HTTP header of the request. For more information see Authentication, Authorization, and Security topic (Home > Authentication, Authorization, and Security) in the Cisco Evolved Programmable Network Manager API documentation.

For Cisco EPN Manager, access to the API is controlled by the user groups such as NBI Read and NBI Write. Each of these groups controls access to a different set of APIs. You can assign a user to multiple groups if you wish. You can check the documentation page of an API resource to determine which user group is required to access it.

**Note**    We recommended to use JSESSIONID during the production environment.

# Getting Started with Cisco EPN Manager REST API

The Cisco Evolved Programmable Network Manager REST API allows an application to interact with Cisco Evolved Programmable Network Manager, programmatically. These requests provide access to resources in Cisco Evolved Programmable Network Manager. With an API call, you can execute Cisco Evolved Programmable Network Manager workflows and Monitor Alarms and Events, Collect device inventory, monitor network clients and usage, configure devices, Device Inventory and so on. For more information see,

the Getting Started topic (**Home** > **Getting Started**) in the Cisco Evolved Programmable Network Manager API documentation.

# REST API Basics and Functional Areas

Cisco Evolved Programmable Network Manager's REST implementation uses multiple calls and filters. For example, when retrieving statistics, the first call provides URLs for the subsequent calls. Those then can be used for retrieving finer detail. In general, the URL parameters become more complicated with each call as more detail is added.

**REST API Basics**

- HTTPS Headers: The following HTTP Headers are used to control the way in which data is returned to a client.

    - Accept

    - Accept-Language

    - Content-Type

    - Accept-Encoding

    - Content-Encoding

- **Query Parameters:**

The API supports query parameters for almost all requests. The following table describes the General REST Query Parameters:

*Table 2: General REST Query Parameters*

| Query Parameter | Applicability | Meaning |
|---|---|---|
| .json | /api/* | When present indicates that the response should be returned in json format. |
| .xml | /api/* | When present indicates that the response should be returned in xml format. This is the default in the absence of the .json parameter. |
| _docs | /api/* | Displays the documentation page. |
| .full | /api/v1/data/T | When 'true' indicates that whole objects be returned rather than just IDs of the entities. |
| .group | api/v1/data/T | Filters the response based on content of the group specified by the string value, and the result of any applied operator. |

| Query Parameter | Applicability | Meaning |
|---|---|---|
| .transform | GET/POST | Supplies the logical-name of a transform to be applied immediately before rendering to XML or JSON. |
| .maxresults | GET Paged | The greatest number of results in terms of the heads of instance trees returned. |
| .firstResult | GET Paged | The first result in terms of the heads of instance trees. |
| .strict | /api/v1/data/T | When 'true', the property names used in queries are validated, and an error thrown if appropriate. (The default - ie when strict=false - is to ignore invalid property names with a simple log message). |
| .case_sensitve | /api/v1/data/T | When 'true', the string values used in filter queries are case sensitive. (The default - ie when .case_sensitive=false - is to treat comparisons as case insensitive). |
| .nocount | /api/v1/data/T | When 'true', the "count", "first", and "last" attributes are not included the response. Getting the values of these attributes requires additional time, therefore setting this to 'true' will improve performance. Defaults to 'false'. |
| _ctx.domain | GET | Sets the active domain to be that named in the query parameter value - this is, by default, "sticky" in the sense that once set, the active-domain remains set. The "stickiness" is not RESTful, and so it can be switched off in configuration. |

**REST API Functional Areas**

- Statistics: The Statistics services provide summary, pre-defined statistical information about the system. Some of the resources of statistics are listed below.

    - GET All Border Routers

    - GET All IMEs

    - GET All RCs

- GET All TCAs

- GET All WANInterfaces

- Report Service: The Report service provides operations to discover and run reports. Reports need to be defined in the system prior to access through the API. The following APIs are supported:

  - GET Get Available Report Templates

  - (Deprecated) GET Get a Report

  - GET Get a ZIP Report

  - GET Run a ZIP Report

- CLI Template Configuration: The CLI Template Configuration service allows a CLI configuration template to be applied to one or more target devices. It also provides a way to upload, delete, and get the CLI templates in the system. The following APIs are supported:

  - GET CLI Configuration Templates

  - DELETE Delete Configuration Template

  - DELETE Delete Configuration Template Folder

  - GET Download Configuration Template

  - GET List Configuration Template Folders

  - GET List Configuration Templates

  - GET List Device Types

  - POST Create Configuration Template Folder

  - POST Upload Configuration Template

  - PUT Deploy Configuration Template

  - PUT Deploy Configuration Template Through Job

  - PUT Modify Configuration Template Content

# Statistics

From EPNM you can acquire statistics data in a simple step by step process. This is a fairly consistent general pattern of usage across several REST statistics endpoints. In general, a high-level first call is made that returns a list of available metrics. From that call, a second list of finer statistics for a given metric is returned from which the actual data series is returned.

Following is an example for Statistics - endpoint call succession: ESR PM data for a given circuit:

```
GET /webacs/api/v1/op/statisticsService/circuits?circuitName=VSO5_TO_HUB2
 GET
/webacs/api/v1/op/statisticsService/circuits/metrics?circuitType=ODUUNI&circuitId=161374613
```

```
GET
/webacs/api/v1/op/statisticsService/circuits/metrics/ESR_PM?maxResults=24&timeInterval=6&endpoint-
Name=ODU20/4/0/0/1&location=FEND&circuitType=ODUUNI&deviceId=124606492_10.201.1.174&circuitId=161374613
```

**Note**    The URLs for the last two calls were provided in the body of the previous call.