RESTCONF Programmable Interface

This chapter describes how to set-up and configure an HTTP-based protocol—Representational State Transfer Configuration Protocol (RESTCONF). RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations and events defined in the YANG model.

- Feature Information for RESTCONF Programmable Interface, on page 1
- Prerequisites, on page 2
- Restrictions for RESTCONF Protocol, on page 2
- Information About RESTCONF Programmable Interface, on page 2
- How to Configure RESTCONF Programmable Interface, on page 8
- Configuration Examples for RESTCONF Programmable Interface, on page 9
- Additional References for RESTCONF Programmable Interface, on page 14

Feature Information for RESTCONF Programmable Interface

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.
Table 1: Feature Information for RESTCONF Programmable Interface

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Releases</th>
<th>Feature Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTCONF Programmable</td>
<td>Cisco IOS XE Everest</td>
<td>This chapter describes how to set-up and configure an HTTP-based protocol, representational state transfer configuration protocol (RESTCONF). RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific remote procedure call (RPC) operations and event notifications defined in the YANG model.</td>
</tr>
<tr>
<td>Interface</td>
<td>16.6.1</td>
<td></td>
</tr>
</tbody>
</table>

This feature has been introduced on the ASR 1000 Aggregation Services Routers-ASR1001-HX and ASR1002-HX, CSR 1000v Series Cloud Services Router, and Cisco 4000 Series Integrated Services Routers (ISRs).

The following commands were introduced or modified:

- `ip http server`
- `restconf-yang`

Prerequisites

- Enable the Cisco IOS-HTTP services for RESTCONF. For more information, see Examples for RESTCONF RPCs

Restrictions for RESTCONF Protocol

The following restrictions apply to the RESTCONF protocol:

- Notifications and event streams
- YANG patch
- Optional query parameters, such as, filter, start-time, stop-time, replay, and action
- The RESTCONF feature is not supported on a device running dual IOSd configuration or software redundancy.

Information About RESTCONF Programmable Interface

Overview of RESTCONF

This section describes the protocols and modelling languages that enable a programmatic way of writing configurations to a network device.

- 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.

RESTCONF and NETCONF in IOS

Protocols and Data Models for Programmatic Device
This section describes the protocols and modelling languages that enable a programmatic way of writing configurations to a network device.

• 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.

If a RESTCONF server is co-located with a NETCONF server, then there are protocol interactions with the NETCONF protocol. The RESTCONF server provides access to specific datastores using operation resources, however, the RESTCONF protocol does not specify any mandatory operation resources, each operation resource determine if and how datastores are accessed.

For more information, refer to the Protocols and Data Models for Programmatic Device section in the Catalyst 4500 Series Software Configuration Guide.

HTTPs Methods

The https-based protocol-RESTCONF (RFC 8040), which is a stateless protocol, uses secure HTTP methods to provide CREATE, READ, UPDATE and DELETE (CRUD) operations on a conceptual datastore containing YANG-defined data, which is compatible with a server that implements NETCONF datastores.

The following table shows how the RESTCONF operations relate to NETCONF protocol operations:

<table>
<thead>
<tr>
<th>OPTIONS</th>
<th>SUPPORTED METHODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Read</td>
</tr>
<tr>
<td>PATCH</td>
<td>Update</td>
</tr>
<tr>
<td>PUT</td>
<td>Create or Replace</td>
</tr>
<tr>
<td>POST</td>
<td>Create or Operations (reload, default)</td>
</tr>
<tr>
<td>DELETE</td>
<td>Deletes the targeted resource</td>
</tr>
<tr>
<td>HEAD</td>
<td>Header metadata (no response body)</td>
</tr>
</tbody>
</table>

RESTCONF Root Resource

• A RESTCONF device determines the root of the RESTCONF API through the link element: /well-known/host-meta resource that contains the RESTCONF attribute.
• The RESTCONF device uses the restconf api root resource as the initial part of the path in the request URI.

For example:

Example returning /restconf:

The client might send the following:

GET /.well-known/host-meta HTTP/1.1
Host: example.com
Accept: application/xrd+xml

The server might respond as follows:

HTTP/1.1 200 OK
Content-Type: application/xrd+xml
Content-Length: nnn

<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
<Link rel='restconf' href='/restconf'/>
</XRD>

Example of URIs:

• GigabitEthernet0/0/2 -
  http://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2

• fields=name –
  http://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2?fields=name

• depth=1 -

• Name and IP -

• MTU (fields) -
  https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/?fields=GiagabitEthernet(mtu)

• MTU -

• Port-Channel -

• “Char” to “Hex” conversion chart: http://www.columbia.edu/kermit/ascii.html

**RESTCONF API Resource**

The API resource is the top-level resource located at +restconf. It supports the following media types:

• Application/YANG-Data+XML OR Application/YANG-Data+JSON

• The API resource contains the RESTCONF root resource for the RESTCONF DATASTORE and OPERATION resources. For example:
The client may then retrieve the top-level API resource, using the root resource "/restconf".

```plaintext
GET /restconf HTTP/1.1
Host: example.com
Accept: application/yang-data+json
```

The server might respond as follows:

```plaintext
HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json
```

```json
{
  "ietf-restconf:restconf" : {
    "data" : {},
    "operations" : {},
    "yang-library-version" : "2016-06-21"
  }
}
```

For more information, refer to RFC 3986

## Reserved or Unreserved Characters

**Conbody**

### Methods

The content query parameter controls how descendant nodes of the requested data nodes are processed in the reply:

- Must be supported by the server.
- If not present in URI, the default value is: all. Allowed only for GET/HEAD method.
  
  A "400 Bad Request" status-line is returned if used for other methods or resource types.

Examples for allowed values are:


**Query Parameters (Fields)**

- The depth-query parameter is used to limit the depth of subtrees returned by the server.
- The value of the "depth" parameter is either an integer between 1 and 65535 or the string "unbounded".
- Supported if present in the capability URI.
- If not present in URI, the default value is: “unbounded”.
- Only allowed for GET/HEAD method.

A 400 Bad Request status-line is returned if used for other methods or resource types.

Examples:
Examples:

- The "fields" query parameter is used to optionally identify data nodes within the target resource to be retrieved in a GET method.
- Supported if present in the capability URI.
  Allowed only for GET/HEAD method.
- A "400 Bad Request" status-line is returned if used for other methods or resource types.
- A value of the "fields" query parameter matches the following rule:

  `fields-expr = path "(" fields-expr ")" / path ";" fields-expr / path path = api-identifier [ "/" path ]`

  1. ";" is used to select multiple nodes.
  2. Parentheses are used to specify sub-selectors of a node. Note that there is no path separator character "/" between a "path" field and a left parenthesis character ".".
  3. "/" is used in a path to retrieve a child node of a node.

Examples:

1. Server module information:

2. Name and IP:
• The "point" query parameter uses to specify the insertion point for a data resource that is being created or moved within an ordered-by user list or leaf-list.

• Must be supported by the server:
  • Only allowed for POST and PUT methods.

  The value of the "point" parameter is a string that identifies the path to the insertion point object. The format is the same as a target resource URI string.

Examples:

PUT:


Query Parameters (with defaults)
The 'with-defaults' query parameter is used to specify how information about default data nodes is returned in response to GET requests on data resources. Default basic-mode in capability is explicit.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report-All</td>
<td>All data nodes are reported</td>
</tr>
<tr>
<td>Trim</td>
<td>Data nodes set to the YANG default are not reported</td>
</tr>
<tr>
<td>Explicit</td>
<td>Data nodes set to the YANG default by the client are reported</td>
</tr>
</tbody>
</table>

• The "point" query parameter uses to specify the insertion point for a data resource that is being created or moved within an ordered-by user list or leaf-list.

Examples:

Sync default settings (error):

Intelligent sync (true):
How to Configure RESTCONF Programmable Interface

Configuring RESTCONF

To configure RESTCONF, perform the steps below:

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# enable</td>
<td>Enter your password if prompted.</td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>restconf-yang</td>
<td>Enables the RESTCONF interface on your network device.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device(config)# restconf-yang</td>
<td>Note After the initial enablement through the CLI, network devices can be managed subsequently through a model-based interface.</td>
</tr>
<tr>
<td>Step 4</td>
<td>exit</td>
<td>Exits global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# exit</td>
<td></td>
</tr>
</tbody>
</table>

Enabling Cisco IOS HTTP Services for RESTCONF

This task is mandatory if you want to use the RESTCONF interface and have opted for the standard configuration method; perform the steps below:

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# enable</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>configure terminal</td>
<td>Enters the global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Device# configure terminal</td>
<td></td>
</tr>
</tbody>
</table>
RESTCONF Programmable Interface

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong> restconf</td>
<td>Enables the RESTCONF interface on your network device.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# restconf</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ip http secure-server</td>
<td>The <code>ip http secure-server</code> command enables a secure HTTP (HTTPS) server.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# ip http secure-server</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end</td>
<td>Exits global configuration mode and enters privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Device(config)# end</td>
<td></td>
</tr>
</tbody>
</table>

Configuration Examples for RESTCONF Programmable Interface

**Example: RESTCONF Programmable Interface**

**GET method (running configuration)**

RESTCONF GET

------------------------

URL:

Headers: {'Accept-Encoding': 'gzip, deflate', 'Accept': 'application/yang-data+json, application/yang-data.errors+json'}

Body:
RESTCONF RESPONSE

------------------------

200

{
  "Cisco-IOS-XE-native:severity": "critical"
}

**GET method**

RESTCONF GET

------------------------

Headers: {'Accept-Encoding': 'gzip, deflate', 'Accept': 'application/yang-data+json, application/yang-data.errors+json'}

Body:

Response:

404 Client Error: Not Found

---

PUT

RESTCONF PUT


Headers: {'Content-type': 'application/yang-data+json', 'Accept': 'application/yang-data+json, application/yang-data.errors+json'}

Body:

{
   "severity": "debugging"
}

RESTCONF RESPONSE

-------------------------------

204

---

PATCH

RESTCONF PATCH

-----------------------


Headers: {'Content-type': 'application/yang-data+json', 'Accept': 'application/yang-data+json, application/yang-data.errors+json'}

Body:

{
   "native": {
      "logging": {
         "monitor": {

```
POST

To create port-channel 1 interface and add multiple configurations to the interface.
If the POST method succeeds, a "201 Created" status-line is returned
and there is no response message-body. A "Location" header field
identifying the child resource that was created MUST be present in
the response in this case.


Headers: {'Content-type': 'application/yang-data+json', 'Accept':
'application/yang-data+json, application/yang-data.errors+json'}

Body:
{
   "Cisco-IOS-XE-native:Port-channel": [
   {
      "name": "1",
      "description": "This is a port-channel interface",
      "delay": 22222,
      "load-interval": 30,
      "mtu": 1501
   }
   ]
Final router config:

csr_dmp_ova#show run interface Port-channel 1
Building configuration...

Current configuration: 183 bytes
!
interface Port-channel1
description This is a port-channel interace
mtu 1501
no ip address
load-interval 30
delay 22222
no negotiation auto
no mop enabled
no mop sysid
end

DELETE
Delete Port-channel interface from running config

RESTCONF RESPONSE
------------------------
201

RESTCONF Programmable Interface
Example: RESTCONF Programmable Interface

RESTCONF Programmable Interface
Example: RESTCONF Programmable Interface
{}

RESTCONF RESPONSE
-------------------------------
204

OPTIONS
Provides header information with the methods supported by the server

RESTCONF OPTIONS
-------------------------------
Headers: {'Accept-Encoding': 'gzip, deflate', 'Accept': 'application/yang-data+json, application/yang-data.errors+json'}
Body:

RESTCONF RESPONSE
-------------------------------
200
>>> resp.headers
{'Server': 'nginx', 'Pragma': 'no-cache', 'Content-Type': 'text/html', 'Connection': 'close', 'Cache-Control': 'private, no-cache, must-revalidate, proxy-revalidate', 'Date': 'Mon, 10 Jul 2017 12:06:29 GMT', 'Allow': 'DELETE, GET, HEAD, PATCH, POST, PUT, OPTIONS', 'Accept-Patch': 'application/yang-data+xml, application/yang-data+json', 'Content-Length': '0'}

HEAD
Headers:
Body:
{
    "logging": {
        "alarm": "critical"
    }
}
RESTCONF RESPONSE

409 Conflict status-line. The error-tag value object already exists is returned if used for other methods or resource types.

409 Client Error: Conflict


Note: Successful Response Messages

200 – Message body is present in the RESTCONF response
204 – Message body is not present in RESTCONF response
201 – Message body is not present in RESTCONF response. It is used with POST operation to show that the targeted resource is created.

Error Response Messages

409 - Client Error: Conflict

Additional References for RESTCONF Programmable Interface

Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>YANG data models for various releases of IOS XE, IOS XR, and NX-OS platforms</td>
<td>To access Cisco YANG models in a developer-friendly way, please clone the GitHub repository, and navigate to the vendor/ciscosubdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.</td>
</tr>
</tbody>
</table>

Standards and RFCs

<table>
<thead>
<tr>
<th>Standard/RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 6020</td>
<td>YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)</td>
</tr>
<tr>
<td>RFC 8040</td>
<td>Representational State Transfer Configuration Protocol (RESTCONF)</td>
</tr>
</tbody>
</table>
Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</td>
<td><a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a></td>
</tr>
</tbody>
</table>
Additional References for RESTCONF Programmable Interface