



RESTCONF Protocol

This chapter describes how to configure the HTTP-based 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.

- [Prerequisites for the RESTCONF Protocol, on page 1](#)
- [Restrictions for the RESTCONF Protocol, on page 1](#)
- [Information About the RESTCONF Protocol, on page 2](#)
- [How to Configure the RESTCONF Protocol, on page 8](#)
- [Configuration Examples for the RESTCONF Protocol, on page 13](#)
- [Additional References for the RESTCONF Protocol, on page 16](#)
- [Feature Information for the RESTCONF Protocol, on page 16](#)

Prerequisites for the RESTCONF Protocol

- Enable the Cisco IOS-HTTP services for RESTCONF. For more information, see [Examples for RESTCONF RPCs](#)

Restrictions for the 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 the RESTCONF Protocol

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.

In releases prior to Cisco IOS XE Fuji 16.8.1, an operational data manager (based on polling) was enabled separately. In Cisco IOS XE Fuji 16.8.1 and later releases, operational data works on platforms running NETCONF (similar to how configuration data works), and is enabled by default. For more information on the components that are enabled for operational data queries or streaming, see the [GitHub](#) repository, and view **-oper* in the naming convention.

HTTPs Methods

The HTTPS-based RESTCONF protocol (RFC 8040), is a stateless protocol that 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:

OPTIONS	SUPPORTED METHODS
GET	Read
PATCH	Update
PUT	Create or Replace
POST	Create or Operations (reload, default)
DELETE	Deletes the targeted resource
HEAD	Header metadata (no response body)

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.
- A RESTCONF device uses the RESTCONF API root resource as the initial part of the path in the request URI.

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 -
<https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2>
- fields=name -
<https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2?fields=name>
- depth=1 -
<https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet?depth=1>
- Name and IP -
<https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet/ip/address/primary/name>
- MTU (fields) -
[https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet\(mtu\)](https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet(mtu))
- MTU -
<https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3/mtu>
- Port-Channel -
<https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/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:



Note Media is the type of YANG formatted RPC that is sent to the RESTCONF server (XML or JSON).

- 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"`.

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

The server might respond as follows:

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

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

For more information, refer to RFC 3986

Methods

Methods are HTTPS operations (GET/PATCH/POST/DELETE/OPTIONS/PUT) performed on a target resource. A YANG-formated RPC invokes a particular method on a given resource that pertains to a target YANG model residing in the RESTCONF server. The uniform resource identifier (URI) acts as a location identification for a given resource, so that the client RESTCONF method can locate that particular resource to take an action specified by an HTTPS method or property.

For more information, see *RFC 8040 - RESTCONF Protocol*

RESTCONF YANG-Patch Support

RESTCONF supports YANG-Patch media type as specified by RFC 8072. A YANG-Patch is an ordered list of edits that are applied to the target datastore by the RESTCONF server. The YANG Patch operation is invoked by the RESTCONF client by sending a Patch method request with a representation using either the media type *application/yang-patch+xml* or *application/yang-patch+json*.

A YANG-Patch is identified by a unique patch-id. A patch is an ordered collection of edits and each edit is identified by an edit-id. It has an edit operation ("create", "delete", "insert", "merge", "move", "replace", or "remove") that is applied to the target resource.

To verify if the RESTCONF YANG-Patch is supported issue the following RESTCONF Get request:

```
$ curl -k -s -u admin:DMIdm1! --location-trusted
"https://10.1.1.1/restconf/data/ietf-restconf-monitoring:restconf-state/capabilities" -X
GET

<capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring"
xmlns:rcmon="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">

<capability>urn:ietf:params:restconf:capability:defaults:1.0?basic-mode=explicit</capability>

<capability>urn:ietf:params:restconf:capability:depth:1.0</capability>
<capability>urn:ietf:params:restconf:capability:fields:1.0</capability>
<capability>urn:ietf:params:restconf:capability:with-defaults:1.0</capability>
```

```
<capability>urn:ietf:params:restconf:capability:filter:1.0</capability>
<capability>urn:ietf:params:restconf:capability:replay:1.0</capability>
```

```
<capability>urn:ietf:params:restconf:capability:yang-patch:1.0</capability>
```

```
<capability>http://tail-f.com/ns/restconf/collection/1.0</capability>
<capability>http://tail-f.com/ns/restconf/query-api/1.0</capability>
</capabilities>
```

This section provides a few RESTCONF YANG-Patch examples.

Add Resource Error

While trying to edit a file, the first edit already exists and an error is reported. The rest of the edits are not attempted because the first edit failed. XML encoding is used in this example

The following example show an add resource request from the RESTCONF client:

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
  <patch-id>add-hostname-patch</patch-id>
  <edit>
    <edit-id>edit1</edit-id>
    <operation>create</operation>
    <target>/hostname</target>
    <value>
      <hostname
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">Cat9K-test</hostname>
      </value>
    </edit>
    <edit>
      <edit-id>edit2</edit-id>
      <operation>create</operation>
      <target>/interface/Loopback=1</target>
      <value>
        <interface xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
          <Loopback>
            <name>1</name>
          </Loopback>
        </interface>
      </value>
    </edit>
  </yang-patch>
```

The following examples shows a JSON response from the RESTCONF server:

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdm1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native" -X PATCH -H "Accept:
application/yang-data+json" -d
'@yang_patch_create_hostname' -H "Content-type: application/yang-patch+xml"
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "add-hostname-patch",
    "edit-status": {
      "edit": [
        {
          "edit-id": "edit1",
          "errors": {
            "error": [
              {
                "error-type": "application",
                "error-tag": "data-exists",
                "error-path": "/Cisco-IOS-XE-native:native/hostname",
```

```

      "error-message": "object already exists: /ios:native/ios:hostname"
    }
  ]
}

```

The following example shows an XML response from the RESTCONF server:

```

Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdm1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native" -X PATCH -H "Accept:
application/yang-data+xml" -d
'@yang_patch_create_hostname' -H "Content-type: application/yang-patch+xml"

<yang-patch-status xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
  <patch-id>add-hostname-patch</patch-id>
  <edit-status>
    <edit>
      <edit-id>edit1</edit-id>
      <errors>
        <error>
          <error-type>application</error-type>
          <error-tag>data-exists</error-tag>
          <error-path>
xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native"/>/ios:native/ios:hostname</error-path>

          <error-message>object already exists: /ios:native/ios:hostname</error-message>
        </error>
      </errors>
    </edit>
  </edit-status>
</yang-patch-status>device:/nobackup/folder1/confd_6313/bin $

```

Add Resource Success

The following example shows an edit request:

```

<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
  <patch-id>add-Loopback-patch</patch-id>
  <edit>
    <edit-id>edit1</edit-id>
    <operation>create</operation>
    <target>/Loopback=1</target>
    <value>
      <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <name>1</name>
      </Loopback>
    </value>
  </edit>
</yang-patch>

```

The following example shows that the edit request is successful:

```

Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdm1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
application/yang-data+json"
-d '@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {

```

```

    "patch-id": "add-Loopback-patch",
    "ok" : [null]
  }
}

```

Insert List Entry

The following example shows that the Loopback 1 is inserted after Loopback 0:

```

<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
  <patch-id>insert-Loopback-patch</patch-id>
  <edit>
    <edit-id>edit1</edit-id>
    <operation>insert</operation>
    <target>/Loopback=1</target>
    <point>/Loopback=0</point>
    <where>after</where>
    <value>
      <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <name>1</name>
      </Loopback>
    </value>
  </edit>
</yang-patch>

```

The following example shows that the insert list request is successful:

```

Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdml! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
application/yang-data+json" -d
'@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "insert-Loopback-patch",
    "ok" : [null]
  }
}

```

Move List Entry

The following example shows Loopback 1 is moved before Loopback 0:

```

<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
  <patch-id>move-Loopback-patch</patch-id>
  <edit>
    <edit-id>edit1</edit-id>
    <operation>move</operation>
    <target>/Loopback=1</target>
    <point>/Loopback=0</point>
    <where>before</where>
    <value>
      <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <name>1</name>
      </Loopback>
    </value>
  </edit>
</yang-patch>

```

The following example shows that the move request is successful:

```

Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
application/yang-data+json" -d
'@yang_patch_create_loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "move-Loopback-patch",
    "ok" : [null]
  }
}

```

How to Configure the RESTCONF Protocol

Authentication of NETCONF/RESTCONF Using AAA

Before you begin

NETCONF and RESTCONF connections must be authenticated using authentication, authorization, and accounting (AAA). As a result, RADIUS or TACACS+ users defined with privilege level 15 access are allowed access into the system.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **aaa group server radius** *server-name*
5. **server-private** *ip-address* **key** *key-name*
6. **ip vrf forwarding** *vrf-name*
7. **exit**
8. **aaa authentication login default group** *group-name* **local**
9. **aaa authentication login** *list-name* **none**
10. **aaa authorization exec default group** *group-name* **local**
11. **aaa session-id** **common**
12. **line console** *number*
13. **login authentication** *authentication-list*
14. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode <ul style="list-style-type: none"> • Enter your password if prompted.

	Command or Action	Purpose
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	aaa new-model Example: Device(config)# aaa new-model	Enables AAA.
Step 4	aaa group server radius <i>server-name</i> Example: Device(config)# aaa group server radius ISE	Adds the RADIUS server and enters server group RADIUS configuration mode. <ul style="list-style-type: none"> • The <i>server-name</i> argument specifies the RADIUS server group name.
Step 5	server-private <i>ip-address</i> key <i>key-name</i> Example: Device(config-sg-radius)# server-private 172.25.73.76 key Cisco123	Configures a IP address and encryption key for a private RADIUS server.
Step 6	ip vrf forwarding <i>vrf-name</i> Example: Device(config-sg-radius)# ip vrf forwarding Mgmt-intf	Configures the virtual routing and forwarding (VRF) reference of a AAA RADIUS or TACACS+ server group.
Step 7	exit Example: Device(config-sg-radius)# exit	Exits server group RADIUS configuration mode and returns to global configuration mode.
Step 8	aaa authentication login default group <i>group-name</i> local Example: Device(config)# aaa authentication login default group ISE local	Sets the specified group name as the default local AAA authentication during login.
Step 9	aaa authentication login <i>list-name</i> none Example: Device(config)# aaa authentication login NOAUTH none	Specifies that no authentication is required while logging into a system.
Step 10	aaa authorization exec default group <i>group-name</i> local Example: Device(config)# aaa authorization exec default group ISE local	Runs authorization to determine if an user is allowed to run an EXEC shell.
Step 11	aaa session-id common Example: Device(config)# aaa session-id common	Ensures that session identification (ID) information that is sent out for a given call will be made identical.

	Command or Action	Purpose
Step 12	line console <i>number</i> Example: Device(config)# line console 0	Identifies a specific line for configuration and enter line configuration mode.
Step 13	login authentication <i>authentication-list</i> Example: Device(config-line)# login authentication NOAUTH	Enables AAA authentication for logins.
Step 14	end Example: Device(config-line)# end	Exits line configuration mode and returns to privileged EXEC mode.

Enabling Cisco IOS HTTP Services for RESTCONF

Perform this task to use the RESTCONF interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **restconf**
4. **ip http secure-server**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	restconf Example: Device(config)# restconf	Enables the RESTCONF interface on your network device.
Step 4	ip http secure-server Example: Device(config)# ip http secure-server	Enables a secure HTTP (HTTPS) server.

	Command or Action	Purpose
Step 5	end Example: Device(config)# end	Exits global configuration mode and enters privileged EXEC mode

Verifying RESTCONF Configuration

When a device boots up with the startup configuration, the *nginx* process will be running. However, DMI processes are not enabled.

The following sample output from the **show platform software yang-management process monitor** command shows that the *nginx* process is running:

```
Device# show platform software yang-management process monitor

COMMAND          PID S   VSZ   RSS %CPU %MEM   ELAPSED
nginx             27026 S 332356 18428 0.0 0.4    01:34
nginx             27032 S 337852 13600 0.0 0.3    01:34
```

NGINX is an internal webserver that acts as a proxy webserver. It provides Transport Layer Security (TLS)-based HTTPS. RESTCONF request sent via HTTPS is first received by the NGINX proxy web server, and the request is transferred to the confd web server for further syntax/semantics check.

The following sample output from the **show platform software yang-management process** command shows the status of the all processes when a device is booted with the startup-configuration:

```
Device# show platform software yang-management process

confd             : Not Running
nesd              : Not Running
syncfd           : Not Running
ncsshd           : Not Running
dmiauthd         : Not Running
nginx            : Running
ndbmand          : Not Running
pubd             : Not Running
```

The *nginx* process gets restarted and DMI process are started, when the **restconf** command is configured.

The following sample output from the **show platform software yang-management process** command shows that the *nginx* process and DMI processes are up and running:

```
Device# show platform software yang-management process

confd             : Running
nesd              : Running
syncfd           : Running
ncsshd           : Not Running ! NETCONF-YANG is not configured, hence ncsshd process is
in not running.
dmiauthd         : Running
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running ! nginx process is up due to the HTTP configuration, and it is
restarted when RESTCONF is enabled.
ndbmand          : Running
```

The following sample output from the show platform software yang-management process monitor command displays detailed information about all processes:

```
Device# show platform software yang-management process monitor
```

```
COMMAND          PID S   VSZ   RSS %CPU %MEM   ELAPSED
confd             28728 S 860396 168496 42.2  4.2    00:12
confd-startup.s  28448 S  19664  4496  0.2  0.1    00:12
dmiauthd         29499 S 275356 23340  0.2  0.5    00:10
ndbmand         29321 S 567232 65564  2.1  1.6    00:11
nesd             29029 S 189952 14224  0.1  0.3    00:11
nginx           29711 S 332288 18420  0.6  0.4    00:09
nginx           29717 S 337636 12216  0.0  0.3    00:09
pubd            28237 S 631848 68624  2.1  1.7    00:13
syncfd         28776 S 189656 16744  0.2  0.4    00:12
```

After AAA and the RESTCONF interface is configured, and nginx process and relevant DMI processes are running; the device is ready to receive RESTCONF requests.

Use the **show netconf-yang sessions** command to view the status of NETCONF/RESTCONF sessions:

```
Device# show netconf-yang sessions
```

```
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore
```

```
Number of sessions : 1
```

session-id	transport	username	source-host	global-lock
19	netconf-ssh	admin	2001:db8::1	None

Use the **show netconf-yang sessions detail** command to view detailed information about NETCONF/RESTCONF sessions:

```
Device# show netconf-yang sessions detail
```

```
R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore
```

```
Number of sessions : 1
```

```
session-id      : 19
transport       : netconf-ssh
username        : admin
source-host     : 2001:db8::1
login-time      : 2018-10-26T12:37:22+00:00
in-rpcs         : 0
in-bad-rpcs     : 0
out-rpc-errors  : 0
out-notifications : 0
global-lock     : None
```

Configuration Examples for the RESTCONF Protocol

Example: Configuring the RESTCONF Protocol

RESTCONF Requests (HTTPS Verbs):

The following is a sample RESTCONF request that shows the HTTPS verbs allowed on a targeted resource. In this example, the **logging monitor** command is used..

```
root:~# curl -i -k -X "OPTIONS"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
\
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:27:57 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Allow: DELETE, GET, HEAD, PATCH, POST, PUT, OPTIONS >>>>>>>>> Allowed methods
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Accept-Patch: application/yang-data+xml, application/yang-data+json
Pragma: no-cache

root:~#
```

POST (Create) Request

The POST operation creates a configuration which is not present in the targeted device.



Note Ensure that the **logging monitor** command is not available in the running configuration.

The following sample POST request uses the **logging monitor alerts** command.

```
Device:~# curl -i -k -X "POST"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor" \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
>      -d '${
>      "severity": "alerts"
>    }'
HTTP/1.1 201 Created
Server: nginx
Date: Mon, 23 Apr 2018 14:53:51 GMT
Content-Type: text/html
Content-Length: 0
Location:
https://10.85.116.30/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:53:51 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
```

```
Etag: 1524-495231-97239
Pragma: no-cache

Device:~#
```

PUT: (Create or Replace) Request:

If the specified command is not present on the device, the POST request creates it ; however, if it is already present in the running configuration, the command will be replaced by this request.

The following sample PUT request uses the **logging monitor warnings** command.

```
Device:~# curl -i -k -X "PUT"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
\
>   -H 'Content-Type: application/yang-data+json' \
>   -H 'Accept: application/yang-data+json' \
>   -u 'admin:admin' \
>   -d ${
>   "severity": "warnings"
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 14:58:36 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:57:46 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495466-326956
Pragma: no-cache

Device:~#
```

PATCH: (Update) Request

The following sample PATCH request uses the **logging monitor informational** command.

```
Device:~# curl -i -k -X "PATCH"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native" \
>   -H 'Content-Type: application/yang-data+json' \
>   -H 'Accept: application/yang-data+json' \
>   -u 'admin:admin' \
>   -d ${
>   "native": {
>     "logging": {
>       "monitor": {
>         "severity": "informational"
>       }
>     }
>   }
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:07:56 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:07:56 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-496076-273016
```

```
Pragma: no-cache
Device:~#
```

GET Request (To Read)

The following sample GET request uses the **logging monitor informational** command.

```
Device:~# curl -i -k -X "GET"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
\
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:10:59 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:severity": "informational"
}
Device:~#
```

DELETE Request (To Delete the Configuration)

```
Device:~# curl -i -k -X "DELETE"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
\
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:26:05 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:26:05 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-497165-473206
Pragma: no-cache

linux_host:~#
```

Additional References for the RESTCONF Protocol

Related Documents

Related Topic	Document Title
YANG data models for various releases of IOS XE, IOS XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the GitHub repository, and navigate to the vendor/cisco subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

Standards and RFCs

Standard/RFC	Title
RFC 6020	YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)
RFC 8040	RESTCONF Protocol
RFC 8072	YANG Patch Media Type

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	https://www.cisco.com/c/en/us/support/index.html

Feature Information for the RESTCONF Protocol

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 the RESTCONF Protocol

Feature Name	Releases	Feature Information
RESTCONF Protocol	Cisco IOS XE Everest 16.6.1	<p>RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific RPC operations and event notifications defined in the YANG model.</p> <p>This feature was introduced on the following platforms:</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Router • Cisco ASR 1000 Aggregation Services Routers • Cisco Cloud Services Router 1000V Series <p>The following commands were introduced or modified: ip http server and restconf</p>
	Cisco IOS XE Fuji 16.8.1a	<p>In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco 1000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Fuji 16.9.2	<p>In Cisco IOS XE Fuji 16.9.2, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.11.1	

Feature Name	Releases	Feature Information
		<p>In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controllers • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9800-L Wireless Controllers.
	Cisco IOS XE Amsterdam 17.3.1	<p>In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms
	Cisco IOS XE Bengaluru 17.4.1	In Cisco IOS XE Bengaluru 17.4.1, this feature was implemented on Cisco Catalyst 8000V Edge Software.

Feature Name	Releases	Feature Information
RESTCONF YANG-Patch Support	Cisco IOS XE Amsterdam 17.1.1	<p>RESTCONF supports YANG-Patch media type as specified by RFC 8072.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1000-RP2, ASR1000-RP3, ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X) • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

