

# RESTCONF Agent


- About the RESTCONF Agent, on page 1
- Guidelines and Limitations, on page 2
- Configuring the RESTCONF Agent, on page 2
- Manage a RESTCONF Session, on page 3
- RESTCONT Get / Set, on page 3
- RESTCONF Device YANG RPC, on page 5
- Troubleshooting the RESTCONF Agent, on page 7


# About the RESTCONF Agent

Cisco NX-OS RESTCONF is a HTTP -based protocol for configuring data that are defined in YANG version 1, using datastores defined in NETCONF.

NETCONF defines configuration datastores and a set of Create, Retrieve, Update, and Delete (CRUD) operations that can be used to access these datastores. The YANG language defines the syntax and semantics of datastore content, operational data, protocol operations, and event notifications.

Cisco NX-OS RESTCONF uses HTTP operations to provide CRUD operations on a conceptual datastore containing YANG-defined data. This data is compatible with a server which implements NETCONF datastores.

The RESTCONF protocol supports both XML and JSON payload encodings. User authentication is done through the HTTP Basic Authentication.

The following table shows the Protocol operations that the Cisco NX-OS RESTCONF Agent supports:

***Table 1: Supported Operations***

| RESTCONF | NETCONF Equivalent |
|---|---|
| OPTIONS | NETCONF: none |
| HEAD | NETCONF: none |
| GET | NETCONF: <get-config>, <get> |
| POST | NETCONF: <edit-config> (operation="create") |
| PUT | NETCONF: <edit-config> (operation="create/replace") |
| PATCH | NETCONF: <edit-config> (operation="merge") |

| | |
|---|---|
| DELETE | NETCONF: <edit-config> (operation="delete") |

# Guidelines and Limitations

The RESTCONF Agent has the following guideline and limitation:

- Cisco NX-OS RESTCONF is based on an RFC draft entitled RESTCONF Protocol. For more details, see RESTCONF Protocol draft-ietf-netconf-restconf-10.
- Beginning with Cisco NX-OS Release 10.4(3)F, the following items in RFC 8040, https://datatracker.ietf.org/doc/html/rfc8040, are supported:
  - 3.2 RESTCONF Media Types
  - 4.2 HEAD
- For TLS, v1.3 is supported, and the minimally supported version is v1.2.
- Beginning with Cisco NX-OS Release 10.4(3)F, RETCONF is supported on 92348GC-X.

# Configuring the RESTCONF Agent

This procedure describes how to enable and configure the RESTCONF.

### Before you begin

Before communicating with the switch using RESTCONF, the RESTCONF Agent must be enabled. The RESTCONF Agent is enabled or disabled by entering the **[no] feature restconf** command. **feature nxapi** is needed to run the http server.

## SUMMARY STEPS

1. **configure terminal**
2. (Optional) **feature nxapi**
3. (Optional) **nxapi http port port-num**
4. (Optional) **nxapi https port port-num**
5. **feature restconf**

## DETAILED STEPS

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal` | Enter global configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | (Optional) **feature nxapi**<br>**Example:**<br>`switch(config)# feature nxapi` | Enable nxapi for the HTTP server. |
| **Step 3** | (Optional) **nxapi http port port-num**<br>**Example:**<br>`switch(conf-tm-sub)# nxapi http port 80` | Specifies the port used for. The default port is 80. |
| **Step 4** | (Optional) **nxapi https port port-num**<br>**Example:**<br>`switch(conf-tm-sub)# nxapi https port 443` | Specifies the port used for https. The default port is 443. |
| **Step 5** | **feature restconf**<br>**Example:**<br>`switch(conf-tm-sensor)# feature restconf` | Enable restconf. |

# Manage a RESTCONF Session

RESTCONF runs on top of HTTP. The NX-API server on the switch listens on the port 80 / 443 (or configured port) of the management port IP address. The client can establish a connection with the HTTP server, and RESTCONF subsystem hooks to the "/restconf" URL.

The client can maintain a long-lived HTTP connection to interact with the RESTCONF agent, though it is usually easier to send a single request then terminate. Please refer to the below sections for explanations.

# RESTCONT Get / Set

The following shows an example request using the linux **curl** command.

#### <GET>

This operation retrieves configuration data from a specified datastore.

The following is the example of HTTP Get request and the response message.

```
client-host % curl -X GET -H "Authorization: Basic YWRtaW46Y2lzY28=" -H "Accept:
application/yang.data+xml"
"http://192.0.20.123/restconf/data/Cisco-NX-OS-device:System/bgp-items/inst-items/dom-items/Dom-list?content=config"
-i
HTTP/1.1 200 OK
Server: nginx/1.7.10
Date: Tue, 27 Sep 2016 20:26:03 GMT
Content-Type: application/yang.data+xml
Content-Length: 395
Connection: keep-alive
Set-Cookie: nxapi_auth=admin:147500856185650327
Status: 200 OK
    <Dom-list>
        <name>default</name>
```

```
        <always>enabled</always>
        <bestPathIntvl>300</bestPathIntvl>
        <holdIntvl>180</holdIntvl>
        <kaIntvl>60</kaIntvl>
        <maxAsLimit>0</maxAsLimit>
        <pfxPeerTimeout>30</pfxPeerTimeout>
        <pfxPeerWaitTime>90</pfxPeerWaitTime>
        <reConnIntvl>60</reConnIntvl>
        <rtrId>2.2.2.2</rtrId>
    </Dom-list>
 client-host %
```

### <POST>

This operation writes a specified configuration to the target datastore.

The following is the example of HTTP POST request and the response message.

```
client-host % curl -X POST -H "Authorization: Basic YWRtaW46Y2lzY28=" -H "Content-Type:
application/yang.data+xml" -d '<always>enabled</always><rtrId>2.2.2.2</rtrId>'
"http://192.0.20.123/restconf/data/Cisco-NX-OS-device:System/bgp-items/inst-items/dom-items/Dom-list=default"
-i
HTTP/1.1 201 Created
Server: nginx/1.7.10
Date: Tue, 27 Sep 2016 20:25:31 GMT
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: nxapi_auth=admin:14750085316974134
Status: 201 Created
Location: /System/bgp-items/inst-items/dom-items/Dom-list=default/always/rtrId/
```

### Retrieve Ephemeral Data

Ephemeral data is high volume data. DME provides a batching mechanism to retrieve the data so that each batch is of a manageable size in terms of memory usage. The size of the batch is the number of MOs to be retrieved.

You can find information about which data is ephemeral by the comment "Ephemeral data" in the published Cisco-NX-OS-device.yang file.

The output from ephemeral data is returned, if and only if the RESTCONF URI in the request points to:

- A leaf from ephemeral data
- A container or list with ephemeral data children
- An empty container that is used to wrap a list that has direct ephemeral data children
- System level GET queries do not return ephemeral data.

### Example

This is an example for retrieving ephemeral data.

The client might send the following GET request message.

```
GET
/restconf/data/Cisco-NX-OS-device:System//urib-items/table4-items/Table4-list=management/route4-items
 HTTP/1.1
Host: example.com
Accept: application/yang.data+json
```

The server might respond:

```
HTTP/1.1 200 OK
Date: Fri, 06 Mar 2020 11:10:30 GMT
Server: nginx/1.7.10
Content-Type: application/yang.data+json
{
"route4-items": {
"Route4-list": [{
"prefix": "172.23.167.255/32", "flags": "0", ...
```

# RESTCONF Device YANG RPC

## About Operational Commands in RESTCONF

This feature provides ways to perform model driven operation commands execution on the switch.

The following is the list of supported execution RPCs. Information about the RPCs can be found in the published Cisco-NX-OS-device.yang file.

***Table 2: Device YANG RPC***

| Operation | Device YANG RPC | CLI |
|---|---|---|
| Checkpoint | checkpoint | checkpoint <name><br>checkpoint <file> |
| Rollback | rollback | rollback running-config checkpoint <<br>rollback running-config checkpoint < |
| Install | install_all_nxos<br>install_add<br>install_activate<br>install_deactivate<br>install_commit<br>install_remove | install all nxos <image><br>install {add \| activate \| deactivate \| co |
| Import Crypto Certificate | import_ca_certificate | crypto ca import <trustpoint> pkcs12 |
| Switch Reload or Module Reload | reload | reload [timer <seconds>]<br>reload module <module number> |
| Copy File | copy | copy <source> <destination> |

## Device YANG RPC Examples

### Creating checkpoint

The client might send the following POST request message:

```
POST /restconf/operations/Cisco-NX-OS-device:checkpoint
Accept: application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
"name": "checkpoint-1",
"description": "testing checkpoint through Restconf" }
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

### Rollback

The client might send the following POST request message:

```
POST /restconf/operations/Cisco-NX-OS-device:rollback
Accept: application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
"name": "checkpoint-1",
"action": "create”
}
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

### Install

The client might send the following POST request message:

```
POST /restconf/operations/Cisco-NX-OS-device:install_all_nxos
Accept: application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
"nxos": "bootflash:nxos.10.1.1-jcco.bin" }
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

### Import ca certificate RPC

The client might send the following POST request message:

```
POST /restconf/operations/Cisco-NX-OS-device:import_ca_certificate Accept:
application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
"trustpoint": "mytrustpoint",
"pkcs12": "bootflash:server.pfx",
"passphrase": "mypassphrase"
}
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

### Switch reload

The client might send the following POST request message:

```
POST /restconf/operations/Cisco-NX-OS-device:reload
Accept: application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
}
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

### Module reload

The client might send the following POST request message:

```
The client might send the following POST request message:
POST /restconf/operations/Cisco-NX-OS-device:reload
Accept: application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
"module": "31"
}
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

### Copy file RPC

The client might send the following POST request message:

```
POST /restconf/operations/Cisco-NX-OS-device:reload
Accept: application/yang.operation+json,application/yang.errors+json
Content-type: application/yang.operation+json
Body: {
"input": {
"source": "tftp://10.1.1.1/users/myname/config1.txt",
"destination": "bootflash:",
"vrf": "management"
}
}
```

The server might respond:

```
HTTP/1.1 204 No content
```

# Troubleshooting the RESTCONF Agent

### Check Feature Status

- In Cisco NX-OS, enter the show feature | inc restconf command to check the agent config.
- To view the status of the RESTCONF agent, use the **show feature** command and include the expression restconf.

```
switch-1# show feature | grep restconf
restconf 1 enabled
switch-1#
```

### Check NX-API Connectivity

- Enable the web server by issuing the **feature nxapi** command.
- Ensure that the **nxapi http port 80** command is configured to open the port for HTTP.
- Ensure that the **nxapi https port 443** command is configured to open the port for HTTPS.
- Ping the management port of the switch to verify that the switch is reachable.

### Check Restconf Errors

The following shows a common error message and offers guidelines for resolving it.

- If you receive this message soon after sending a request (for example, seconds), verify the following:

  **Error Message:** *Sorry, the page you are looking for is currently unavailable*

  - The NXAPI feature is enabled as documented in "Troubleshooting Connectivity".
  - The RESTCONF feature is enabled (**show feature | grep restconf**). If RESTCONF is not enabled, enable it (**feature restconf**).
  - The port is configured for HTTP or HTTPS by NX-API. Use show nxapi to verify that the port is configured.

    ```
    switch-1# show nxapi
    nxapi enabled
    HTTP Listen on port 80
    HTTPS Listen on port 443
    ```

    If the port is not configured for HTTP or HTTPs, configure it by issuing **nxapi http port 80** or **nxapi https port 443**.

- If you receive this message long after sending a request (for example, minutes), ensure that the system is not overloaded with excessive concurrent requests to query from the top level of the switch. Excessive top-level queries can create a significant resource burden.

  You can ensure the switch is not overloaded by either of the following:

  - Throttle back the number of requests that the client is sending.
  - On the switch, restart the RESTCONF agent by issuing **no feature restconf**, then **feature restconf**.

### Accounting Log for RESTCONF Agent

For write operations such as POST, PUT, PATCH, or DELETE, RESTCONF would emit the relevant accounting log. It would include both the original received request, as well as the eventual changes applied to the switch.

You can see the accounting log using the show accounting log command.

Consider the following example request:

```
---
curl -s -L --request POST --user admin: --header 'Content-Type: application/yang.data+json'
 --url "restconf/data/Cisco-NX-OS-device:System/intf-items/lb-items/LbRtdIf-list=lo10"
--data-raw @request.txt
Payload:
<descr>test</descr>
Or
{"descr":"test"}
---
```

The accounting log shall include the following items:

***Table 3: Changes applied to the switch***

| Item | Description |
|---|---|
| Context | Session ID and user |
| Operation | COMMIT/ABORT |
| Database | Running or Candidate |
| ConfigMO | MO tree's text representation. Up to 3K characters. |
| Status | SUCCESS/FAILED |

Example:

```
Wed Jun 29 13:53:37
2022:type=update:id=3180018864:user=admin:cmd=(COMMIT),database=[running],configMo=[
<topSystem childAction="" dn="sys" status="created,modified"><interfaceEntity childAction=""
rn="intf" status="created,modified"><l3LbRtdIf childAction="" descr="test" id="lo10" rn="lb-
[lo10]" status="created,modified"/></interfaceEntity></topSystem>] (SUCCESS)
```

***Table 4: Original received request***

| Item | Description |
|---|---|
| Context | Session ID and user |
| Operation | RESTCONF:POST, RESTCONF:PUT, RESTCONF:PATCH, RESTCO |
| Source IP | RESTCONF Client IP |
| URL | HTTP URL |
| Payload | Received XML/JSON Request. Up to 3K characters. |
| Status | SUCCESS/FAILED |

Example:

```
Wed Jun 29 13:53:37
2022:type=update:id=3180018864:user=admin:cmd=(RESTCONF:POST),sourceIp=[192.168.1.2],
url=[/restconf/data/Cisco-NX-OS-device:System/intf-items/lb-items/LbRtdIflist=lo10],payload=[<descr>test</descr>]
 (SUCCESS)
```

In case of failed request, based on the failed scenarios, a user may not observe both the logs.

**Invalid request:**

The invalid request would be rejected without making a configuration change, thus only the original request would be logged.

Example:

```
Wed Jun 29 20:16:26
2022:type=update:id=3180018864:user=admin:cmd=(RESTCONF:POST),
sourceIp=[192.168.1.2],url=[/restconf/data/Cisco-NX-OS-device:System/intf-items/lb-items/LbRtdIf-list=lo10],payload=[<descr>test</descr>]
 (FAILED)
```

**Request fails due to various configuration restrictions:**

In this case, both the failed configuration attempt and the original request would be logged.

Example:

```
Wed Jun 29 20:32:01
2022:type=update:id=3180018864:user=admin:cmd=(COMMIT),database=[running],
configMo=[<topSystem childAction="" dn="sys" status="created,modified"><telemetryEntity
childAction="" rn="tm" status="created,modified"><telemetryCertificate childAction=""
filename="foo" hostname="foo" rn="certificate" status="created,modified"
trustpoint="test"/></telemetryEntity></topSystem>] (FAILED)
Wed Jun 29 20:32:01
2022:type=update:id=3180018864:user=admin:cmd=(RESTCONF:PATCH),
sourceIp=[192.168.1.2],url=[/restconf/data/Cisco-NX-OS-device:System/tm-items/certificate-items],payload=[<trustpoint>test</trustpoint><hostname>foo</hostname><filename>
 foo</filename>] (FAILED)
```