# NX-API REST

This chapter contains the following topics:



# About NX-API REST

### NX-API REST

On Cisco Nexus switches, configuration is performed using command-line interfaces (CLIs) that run only on the swtich. NX-API REST improves the accessibility of the Cisco Nexus configuration by providing HTTP/HTTPS APIs that:

- Make specific CLIs available outside of the switch.
- Enable configurations that would require issuing many CLI commands by combining configuration actions in relatively few HTTP/HTTPS operations.

NX-API REST supports **show** commands, basic and advanced switch configurations, and Linux Bash.

NX-API REST uses HTTP/HTTPS as its transport. CLIs are encoded into the HTTP/HTTPS POST body. The NX-API REST backend uses the Nginx HTTP server. The Nginx process,and all of its children processes, are under Linux cgroup protection where the CPU and memory usage is capped. If the Nginx memory usage exceeds the cgroup limitations, the Nginx process is restarted and the NX-API configuration (the VRF, port, and certificate configurations) is restored.

For more information about the Cisco Nexus 3000 and 9000 Series NX-API REST SDK, see https://developer.cisco.com/docs/nx-os-n3k-n9k-api-ref/.

# DME Config Replace Through REST

## About DME Full Config Replace Through REST Put

Beginning with Cisco NX-OS Release 9.3(1), Cisco NX-OS supports model-based full config replace through REST PUT operations. This method of replacing configurations uses the Cisco DME model.

The DME Full Config replace feature enables you to use the REST programmatic interface to replace the switch running configuration. The feature provides the following extra benefits:DME full config replace occurs through a PUT operation. All parts of the config tree (system-level, subtree, and leaf) support DME full config replace.

- Supports non-disruptive replacement of the switch configuration
- Supports automation
- Offers the ability to selectively modify features without affecting other features or their configs.
- Simplifies config changes and eliminates human error by enabling you to specify the final config outcome. The switch calculates the differences and pushes them to the affected parts of config tree.

**Note** Although not accomplished through a programmatic interface, you can also achieve a full config replace by using the **config replace config-file-name** Cisco NX-OS CLI command.

# Guidelines and Limitations

The following are the guidelines and limitations for the DME full config replace feature:

- For information about supported platforms for Cisco NX-OS prior to release 9.3(x), see Platform Support for Programmability Features. Starting with Cisco NX-OS release 9.3(x), for information about supported platforms, see the Nexus Switch Platform Matrix.
- DME is not supported on N9K-92348GC-X.
- It is important for you to know the tree and know where you are applying the config replace. If you are using the Sandbox for the config replace operation, the Sandbox defaults to the subtree, so you might need to change the URI to target the correct node in the config tree.
- If you use the NX-OS Sandbox to Convert (for Replace), you must use the POST operation because of the presence of the `status: 'replaced'` attribute in the request. If you are using any other conversion option, you can use the PUT operation.
- If you use the REST PUT option for this feature on a subtree node, config replace operation is applied to the entire subtree. The target subtree node is correctly changed with the config replace data in the PUT, but be aware that leaf nodes of the subtree node are also affected by being set to default values.

  If you do not want the leaf nodes to be affected, do not use a PUT operation. Instead, you can use a POST operation with the `status:'replaced'` attribute.

  If you are applying the config replace to a leaf node, the PUT operation operates predictably.

# Replacing Property-Level Configuration Through REST POST

Cisco's DME model supports property-level config replace for CLI-based features through a REST POST operation. You can replace the config for the property of a feature through the NX-OS Sandbox by generating a request payload and sending it to the switch through a REST POST operation. For information about the NX-OS Sandbox, see NX-API Developer Sandbox.

**Step 1** Connect to the switch through NX-OS Sandbox through HTTPS and provide your login credentials.

**Step 2** In the work area, enter the CLI for the feature that you want to change.

**Step 3** In the field below the work area, set the URI to the MO in the tree for the feature that you want to configure. This MO level is where you will send the Put request.

**Step 4** For Method, select `NX-API (DME)`.

**Step 5** For Input Type, select CLI.

**Step 6** From the Convert drop-down list, select `Convert (for replace)` to generate the payload in the Request pane.

**Step 7** Click the request using a **POST** operation to the switch..

**Note** Property-level config replace can fail if the config is a default config because the replace operation tries to delete all the children MOs and reset all properties to default.

# Replacing Feature-Level Config Through REST PUT

Cisco DME supports replacing feature-level configurations through REST PUT operations. You can replace the configuration for specific features by sending a PUT at the feature level of the model.

Use the following procedure:

**Step 1** From the client, issue a REST PUT operation at the model object (MO) of the feature:

a) The Put must specify the URL from the top System level to the MO of the feature.

For example, for a BGP `/api/mo/sys/bgp.json`

The payload must be a valid config, and the config must be retrievable from the switch at any time by issuing a GET on the DN of the feature. For example, for BGP, `/api/mo/sys/bgp.json?rsp-subtree=full&rsp-prop-include=set-config-only`.

b) The payload for the feature should start with the MO that you want to replace (for example, `bgp`).

For example:

```
{
            "bgpInst": {
              "attributes": {
                "asn": "100",
                "rn": "inst"
              },
              "children": [

              ... content removed for brevity ...

                {
                  "bgpDom": {
                    "attributes": {
                      "name": "vrf1",
                      "rn": "dom-vrf1"
                    },
                    "children": [
                      {
                        "bgpPeer": {
                          "attributes": {
```

```
                    "addr": "10.1.1.1",
                    "inheritContPeerCtrl": "",
                    "rn": "peer-[10.1.1.1]"
                  }
                }
              }
            ]
          }
        },
        {
          "bgpDom": {
            "attributes": {
              "name": "default",
              "rn": "dom-default",
              "rtrId": "1.1.1.1"
            }
          }
        }
      ]
    }
  }
```

**Step 2** Send a GET on the DN you used for the config replace by using `/api/mo/sys/bgp.json?rsp-subtree=full&rsp-prop-include=set-config-only`.

**Step 3** (Optional) Compare the payload that you sent with the GET on the DN you replaced. The payload of the GET should be the same as the payload you sent.

# Troubleshooting Config Replace for REST PUT

The following are steps to help troubleshoot if config replace through a REST Put operation is not successful.

**Step 1** Check if the request is valid.

The URL, operation, and payload should be valid. For example, if the URL is `api/mo/sys/foo.json` then the payload should start with `foo`

**Step 2** Make sure the payload is valid and contains only the config properties which are:

- Successfully set
- Taken from a valid device config

To get only the config properties, use a GET that filters for `rsp-subtree=full&rsp-prop-include=set-config-only`

**Step 3** To validate the payload, send it to the switch using a DME POST operation.

**Step 4** Check the error to verify that it has the name of the MO and property.

**Step 5** Validate the payload also has the name of the MO and property.