



NX-API

- [About NX-API, page 1](#)
- [Using NX-API, page 3](#)
- [XML and JSON Supported Commands, page 12](#)

About NX-API

On Cisco Nexus devices, command-line interfaces (CLIs) are run only on the device. NX-API improves the accessibility of these CLIs by making them available outside of the switch by using HTTP/HTTPS. You can use this extension to the existing Cisco Nexus CLI system on the Cisco Nexus 3000 Series devices. NX-API supports **show** commands, configurations, and Linux Bash.

NX-API supports JSON-RPC.

Feature NX-API

- Feature NX-API is required to be enabled for access the device through sandbox.
- `| json` on the device internally uses python script to generate output.
- NX-API can be enabled either on http/https via ipv4 or ipv6

```
BLR-VXLAN-NPT-CR-179# show nxapi
nxapi enabled
HTTP Listen on port 80
HTTPS Listen on port 443
BLR-VXLAN-NPT-CR-179#
```
- NX-API is internally spawning third-party NGINX process, which handler receive/send/processing of http requests/response

```
nxapi certificate {httpsCRT |httpskey}
nxapi certificate enable
```
- NX-API Certificates can be enabled for https
- Default port for nginx to operate is 80/443 for http/https respectively. It can also be changed using the following CLI's

```
nxapi {http|https} port port
```

Transport

NX-API uses HTTP/HTTPS as its transport. CLIs are encoded into the HTTP/HTTPS POST body.

The NX-API 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 restored.

Message Format

**Note**

- NX-API XML output presents information in a user-friendly format.
- NX-API XML does not map directly to the Cisco NX-OS NETCONF implementation.
- NX-API XML output can be converted into JSON.

Security

NX-API supports HTTPS. All communication to the device is encrypted when you use HTTPS.

NX-API is integrated into the authentication system on the device. Users must have appropriate accounts to access the device through NX-API. NX-API uses HTTP basic authentication. All requests must contain the username and password in the HTTP header.

**Note**

You should consider using HTTPS to secure your user's login credentials.

You can enable NX-API by using the **feature** manager CLI command. NX-API is disabled by default.

NX-API provides a session-based cookie, **nxapi_auth** when users first successfully authenticate. With the session cookie, the username and password are included in all subsequent NX-API requests that are sent to the device. The username and password are used with the session cookie to bypass performing the full authentication process again. If the session cookie is not included with subsequent requests, another session cookie is required and is provided by the authentication process. Avoiding unnecessary use of the authentication process helps to reduce the workload on the device.

**Note**

A **nxapi_auth** cookie expires in 600 seconds (10 minutes). This value is a fixed and cannot be adjusted.

**Note**

NX-API performs authentication through a programmable authentication module (PAM) on the switch. Use cookies to reduce the number of PAM authentications, which reduces the load on the PAM.

Using NX-API

The commands, command type, and output type for the Cisco Nexus 3000 Series devices are entered using NX-API by encoding the CLIs into the body of a HTTP/HTTPS POST. The response to the request is returned in XML or JSON output format.



Note For more details about NX-API response codes, see [Table of NX-API Response Codes](#).

You must enable NX-API with the **feature** manager CLI command on the device. By default, NX-API is disabled.

The following example shows how to configure and launch the NX-API Sandbox:

- Enable the management interface.

```
switch# conf t
switch(config)# interface mgmt 0
switch(config)# ip address 198.51.100.1/24
switch(config)# vrf context managment
switch(config)# ip route 203.0.113.1/0 1.2.3.1
```

- Enable the NX-API **nxapi** feature.

```
switch# conf t
switch(config)# feature nxapi
```

The following example shows a request and its response in XML format:

Request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ins_api>
  <version>0.1</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>session1</sid>
  <input>show switchname</input>
  <output_format>xml</output_format>
</ins_api>
```

Response:

```
<?xml version="1.0"?>
<ins_api>
  <type>cli_show</type>
  <version>0.1</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body>
        <hostname>switch</hostname>
      </body>
      <input>show switchname</input>
      <msg>Success</msg>
      <code>200</code>
    </output>
  </outputs>
</ins_api>
```

The following example shows a request and its response in JSON format:

Request:

```
{
  "ins_api": {
    "version": "0.1",
    "type": "cli_show",
    "chunk": "0",
    "sid": "session1",
    "input": "show switchname",
    "output_format": "json"
  }
}
```

Response:

```
{
  "ins_api": {
    "type": "cli_show",
    "version": "0.1",
    "sid": "eoc",
    "outputs": {
      "output": {
        "body": {
          "hostname": "switch"
        },
        "input": "show switchname",
        "msg": "Success",
        "code": "200"
      }
    }
  }
}
```

Using the Management Interface for NX-API calls

It is recommended to use the management interface for NX-API calls.

When using non-management interface and a custom port for NX-API an entry should be made in the CoPP policy to prevent NX-API traffic from hitting the default copp entry which could unfavorably treat API traffic.

**Note**

It is recommended to use the management interface for NX-API traffic. If that is not possible and a custom port is used, the "copp-http" class should be updated to include the custom NX-API port.

The following example port 9443 is being used for NX-API traffic.

This port is added to the copp-system-acl-http ACL to allow it to be matched under the copp-http class resulting on 100 pps policing. (This may need to be increased in certain environments.)

```
!
ip access-list copp-system-acl-http
 10 permit tcp any any eq www
 20 permit tcp any any eq 443
 30 permit tcp any any eq 9443 <-----
!
class-map type control-plane match-any copp-http
 match access-group name copp-system-acl-http
!
policy-map type control-plane copp-system-policy
 class copp-http
  police pps 100
!
```

NX-API Sandbox

The NX-API Sandbox is the web-based user interface that you use to enter the commands, command type, and output type for the Cisco Nexus 3000 Series device using HTTP/HTTPS. After posting the request, the output response is displayed.

By default, NX-API is disabled. Begin enabling NX-API with the **feature** manager CLI command on the switch. Then enable NX-API with the **nxapi sandbox** command.

Use a browser to access the NX-API Sandbox.

**Note**

When using the NX-API Sandbox, Cisco recommends that you use the Firefox browser, release 24.0 or later.

You can also NX-API over IPv6.

The following example shows how to configure and launch the NX-API Sandbox:

- Enable the management interface.

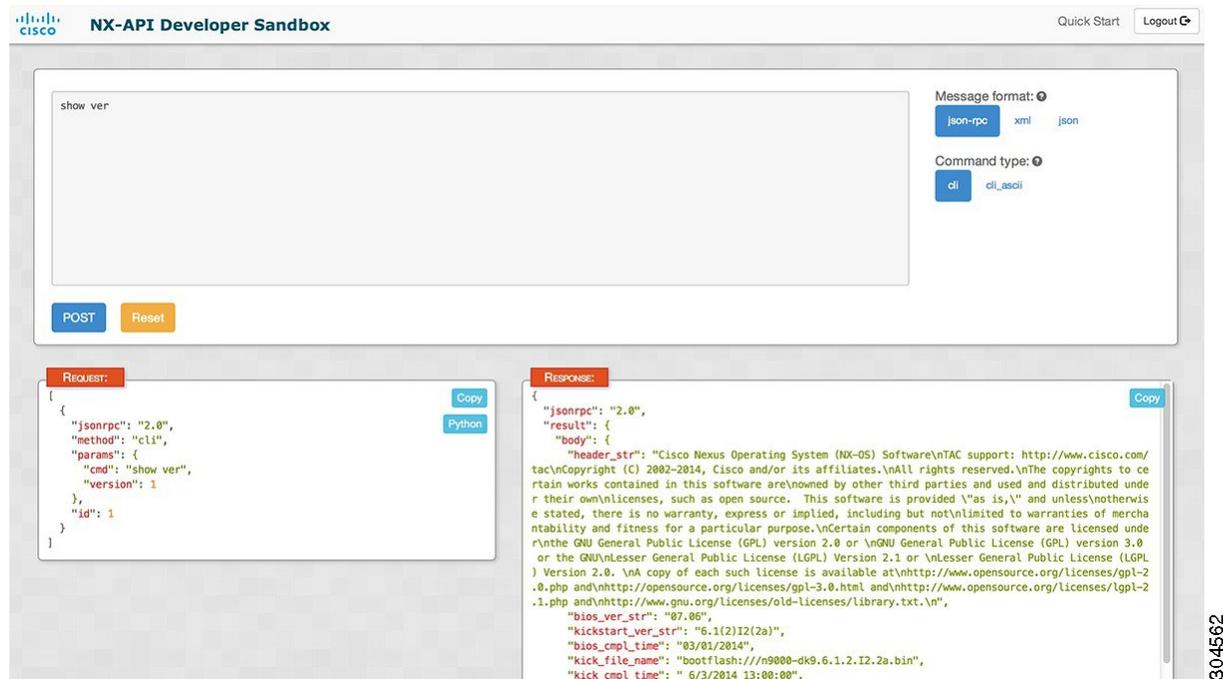
```
switch# conf t
switch(config)# interface mgmt 0
switch(config)# ip address 198.51.100.1/24
switch(config)# vrf context management
switch(config)# ip route 203.0.113.1/0 1.2.3.1
```

- Enable the NX-API **nxapi** feature.

```
switch# conf t
switch(config)# feature nxapi
switch(config)# nxapi sandbox
```

- Open a browser and enter `http://mgmt-ip` to launch the NX-API Sandbox. The following figure is an example of a request and output response.

Figure 1: NX-API Sandbox with Example Request and Output Response



In the NX-API Sandbox, you specify the commands, command type, and output type in the top pane. Click the POST Request button above the left pane to post the request. Brief descriptions of the request elements are displayed below the left pane.

After the request is posted, the output response is displayed in the right pane.

The following sections describe the commands to manage NX-API and descriptions of the elements of the request and the output response.

NX-API Management Commands

You can enable and manage NX-API with the CLI commands listed in the following table.

Table 1: NX-API Management Commands

NX-API Management Command	Description
<code>feature nxapi</code>	Enables NX-API.
<code>no feature nxapi</code>	Disables NX-API.
<code>nxapi {http https} port port</code>	Specifies a port.

NX-API Management Command	Description
no nxapi {http https}	Disables HTTP/HTTPS.
show nxapi	Displays port information.
nxapi certificate { <i>httpsert</i> <i>httpskey</i> }	Specifies the upload of the following: <ul style="list-style-type: none"> • HTTPS certificate when <i>httpsert</i> is specified. • HTTPS key when <i>httpskey</i> is specified.
nxapi certificate <i>enable</i>	Enables a certificate.

NX-API Request Elements

NX-API request elements are sent to the device in XML format or JSON format. The HTTP header of the request must identify the content type of the request.

You use the NX-API elements that are listed in the following table to specify a CLI command:

Table 2: NX-API Request Elements

NX-API Request Element	Description
version	Specifies the NX-API version.

NX-API Request Element	Description
<i>type</i>	<p>Specifies the type of command to be executed.</p> <p>The following types of commands are supported:</p> <ul style="list-style-type: none"> • cli_show CLI show commands that expect structured output. If the command does not support XML output, an error message is returned. • cli_show_ascii CLI show commands that expect ASCII output. This aligns with existing scripts that parse ASCII output. Users are able to use existing scripts with minimal changes. • cli_conf CLI configuration commands. • bash Bash commands. Most non-interactive Bash commands are supported by NX-API. <p>Note</p> <ul style="list-style-type: none"> • Each command is only executable with the current user's authority. • The pipe operation is supported in the output when the message type is ASCII. If the output is in XML format, the pipe operation is not supported. • A maximum of 10 consecutive show commands are supported. If the number of show commands exceeds 10, the 11th and subsequent commands are ignored. • No interactive commands are supported.

NX-API Request Element	Description						
<i>chunk</i>	<p>Some show commands can return a large amount of output. For the NX-API client to start processing the output before the entire command completes, NX-API supports output chunking for show commands.</p> <p>Enable or disable chunk with the following settings:</p> <table border="1" data-bbox="824 459 1515 590"> <tr> <td data-bbox="824 459 938 522">0</td> <td data-bbox="938 459 1515 522">Do not chunk output.</td> </tr> <tr> <td data-bbox="824 522 938 590">1</td> <td data-bbox="938 522 1515 590">Chunk output.</td> </tr> </table> <p>Note Only show commands support chunking. When a series of show commands are entered, only the first command is chunked and returned.</p> <p>The output message format is XML. (XML is the default.) Special characters, such as < or >, are converted to form a valid XML message (< is converted into &lt; > is converted into &gt;).</p> <p>Note You can use XML SAX to parse the chunked output. When chunking is enabled, the message format is limited to XML. JSON output format is not supported when chunking is enabled.</p>	0	Do not chunk output.	1	Chunk output.		
0	Do not chunk output.						
1	Chunk output.						
<i>sid</i>	<p>The session ID element is valid only when the response message is chunked. To retrieve the next chunk of the message, you must specify a <i>sid</i> to match the <i>sid</i> of the previous response message.</p>						
<i>input</i>	<p>Input can be one command or multiple commands. However, commands that belong to different message types should not be mixed. For example, show commands are cli_show message type and are not supported in cli_conf mode.</p> <p>Note Except for bash, multiple commands are separated with " ; ". (The ; must be surrounded with single blank characters.)</p> <p>For bash, multiple commands are separated with ";". (The ; is not surrounded with single blank characters.)</p> <p>The following are examples of multiple commands:</p> <table border="1" data-bbox="824 1518 1515 1751"> <tr> <td data-bbox="824 1518 948 1593">cli_show</td> <td data-bbox="948 1518 1515 1593">show version ; show interface brief ; show vlan</td> </tr> <tr> <td data-bbox="824 1593 948 1669">cli_conf</td> <td data-bbox="948 1593 1515 1669">interface Eth4/1 ; no shut ; switchport</td> </tr> <tr> <td data-bbox="824 1669 948 1751">bash</td> <td data-bbox="948 1669 1515 1751">cd /bootflash;mkdir new_dir</td> </tr> </table>	cli_show	show version ; show interface brief ; show vlan	cli_conf	interface Eth4/1 ; no shut ; switchport	bash	cd /bootflash;mkdir new_dir
cli_show	show version ; show interface brief ; show vlan						
cli_conf	interface Eth4/1 ; no shut ; switchport						
bash	cd /bootflash;mkdir new_dir						

NX-API Request Element	Description		
<i>output_format</i>	The available output message formats are the following:		
	<table border="1"> <tr> <td data-bbox="781 323 1019 384">xml</td> <td data-bbox="1019 323 1482 384">Specifies output in XML format.</td> </tr> </table>	xml	Specifies output in XML format.
	xml	Specifies output in XML format.	
<table border="1"> <tr> <td data-bbox="781 390 1019 451">json</td> <td data-bbox="1019 390 1482 451">Specifies output in JSON format.</td> </tr> </table>	json	Specifies output in JSON format.	
json	Specifies output in JSON format.		
	<p>Note The Cisco Nexus 3000 Series CLI supports XML output, which means that the JSON output is converted from XML. The conversion is processed on the switch.</p> <p>To manage the computational overhead, the JSON output is determined by the amount of output. If the output exceeds 1 MB, the output is returned in XML format. When the output is chunked, only XML output is supported.</p> <p>The content-type header in the HTTP/HTTPS headers indicate the type of response format (XML or JSON).</p>		

NX-API Response Elements

The NX-API elements that respond to a CLI command are listed in the following table:

Table 3: NX-API Response Elements

NX-API Response Element	Description
version	NX-API version.
type	Type of command to be executed.
sid	Session ID of the response. This element is valid only when the response message is chunked.
outputs	<p>Tag that encloses all command outputs.</p> <p>When multiple commands are in <code>cli_show</code> or <code>cli_show_ascii</code>, each command output is enclosed by a single output tag.</p> <p>When the message type is <code>cli_conf</code> or <code>bash</code>, there is a single output tag for all the commands because <code>cli_conf</code> and <code>bash</code> commands require context.</p>
output	<p>Tag that encloses the output of a single command output.</p> <p>For <code>cli_conf</code> and <code>bash</code> message types, this element contains the outputs of all the commands.</p>

NX-API Response Element	Description
input	Tag that encloses a single command that was specified in the request. This element helps associate a request input element with the appropriate response output element.
body	Body of the command response.
code	Error code returned from the command execution. NX-API uses standard HTTP error codes as described by the Hypertext Transfer Protocol (HTTP) Status Code Registry (http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml).
msg	Error message associated with the returned error code.

About JSON (JavaScript Object Notation)

JSON is a light-weight text-based open standard designed for human-readable data and is an alternative to XML. JSON was originally designed from JavaScript, but it is language-independent data format. The JSON/CLI Execution is currently supported in Nexus 3500.



Note

The NX-API/JSON functionality is now available on the Cisco Nexus 3500 Series platform.

The two primary Data Structures that are supported in some way by nearly all modern programming languages are as follows:

- Ordered List :: Array
- Unordered List (Name/Value pair) :: Objects

JSON /XML output for a show command can also be accessed via sandbox.

CLI Execution

Show_Command | json

Example Code

```
BLR-VXLAN-NPT-CR-179# show cdp neighbors | json
{"TABLE_cdp_neighbor_brief_info": {"ROW_cdp_neighbor_brief_info": [{"ifindex": "83886080", "device_id": "SW-SPARSHA-SAVBU-F10", "intf_id": "mgmt0", "ttl": "148", "capability": ["switch", "IGMP_cnd_filtering"], "platform_id": "cisco WS-C2960S-48TS-L", "port_id": "GigabitEthernet1/0/24"}, {"ifindex": "436207616", "device_id": "BLR-VXLAN-NPT-CR-178 (FOC1745R01W)", "intf_id": "Ethernet1/1", "ttl": "166", "capability": ["router", "switch", "IGMP_cnd_filtering", "Supports-STP-Dispute"], "platform_id": "N3K-C3132Q-40G", "port_id": "Ethernet1/1"}]}}
```

XML and JSON Supported Commands

Cisco NX-OS Release 6.0(2)U4(1) introduces support for XML and JSON output of the following commands:

- show bgp all
- show bgp process
- show bgp convergence
- show bgp ip unicast/multicast
- show bgp ipv4 unicast/multicast
- show bgp ipv6 unicast/multicast
- show bgp paths
- show bgp peer-policy
- show bgp vrf
- show bgp sessions
- show bgp statistics
- show consistency-checker forwarding ipv4
- show consistency-checker forwarding ipv6
- show lldp neighbors
- show lldp neighbors detail
- show lldp neighbors interface ethernet x/x
- show lldp neighbors interface ethernet x/x detail
- show lldp portid-subtype
- show lldp timers
- show lldp tlv-select
- show lldp traffic
- show lldp traffic interface ethernet x/x
- show process memory
- show process cpu & show process
- show routing vrf all
- show system internal forwarding route summary
- show system resources

Examples of XML and JSON Output

This example shows how to display the unicast and multicast routing entries in hardware tables in JSON format:

```
switch(config)# show hardware profile status | json
{"total_lpm": ["8191", "1024"], "total_host": "8192", "max_host4_limit": "4096",
 "max_host6_limit": "2048", "max_mcast_limit": "2048", "used_lpm_total": "9", "u
sed_v4_lpm": "6", "used_v6_lpm": "3", "used_v6_lpm_128": "1", "used_host_lpm_tot
al": "0", "used_host_v4_lpm": "0", "used_host_v6_lpm": "0", "used_mcast": "0", "
used_mcast_oif1": "2", "used_host_in_host_total": "13", "used_host4_in_host": "1
2", "used_host6_in_host": "1", "max_ecmp_table_limit": "64", "used_ecmp_table":
"0", "mfib_fd_status": "Disabled", "mfib_fd_maxroute": "0", "mfib_fd_count": "0"
}
switch(config)#
```

This example shows how to display the unicast and multicast routing entries in hardware tables in XML format:

```
switch(config)# show hardware profile status | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://w
ww.cisco.com/nxos:1.0:fib">
  <nf:data>
    <show>
      <hardware>
        <profile>
          <status>
            <_XML_OPT_Cmd_dynamic_tcama_status__readonly__>
              <_readonly__>
                <total_lpm>8191</total_lpm>
                <total_host>8192</total_host>
                <total_lpm>1024</total_lpm>
                <max_host4_limit>4096</max_host4_limit>
                <max_host6_limit>2048</max_host6_limit>
                <max_mcast_limit>2048</max_mcast_limit>
                <used_lpm_total>9</used_lpm_total>
                <used_v4_lpm>6</used_v4_lpm>
                <used_v6_lpm>3</used_v6_lpm>
                <used_v6_lpm_128>1</used_v6_lpm_128>
                <used_host_lpm_total>0</used_host_lpm_total>
                <used_host_v4_lpm>0</used_host_v4_lpm>
                <used_host_v6_lpm>0</used_host_v6_lpm>
                <used_mcast>0</used_mcast>
                <used_mcast_oif1>2</used_mcast_oif1>
                <used_host_in_host_total>13</used_host_in_host_total>
                <used_host4_in_host>12</used_host4_in_host>
                <used_host6_in_host>1</used_host6_in_host>
                <max_ecmp_table_limit>64</max_ecmp_table_limit>
                <used_ecmp_table>0</used_ecmp_table>
                <mfib_fd_status>Disabled</mfib_fd_status>
                <mfib_fd_maxroute>0</mfib_fd_maxroute>
                <mfib_fd_count>0</mfib_fd_count>
              </_readonly__>
            </_XML_OPT_Cmd_dynamic_tcama_status__readonly__>
          </status>
        </profile>
      </hardware>
    </show>
  </nf:data>
</nf:rpc-reply>
]]>]]>
switch(config)#
```

This example shows how to display LLDP timers configured on the switch in JSON format:

```
switch(config)# show lldp timers | json
{"ttl": "120", "reinit": "2", "tx_interval": "30", "tx_delay": "2", "hold_mplier": "4", "notification_interval": "5"}
switch(config)#
```

This example shows how to display LLDP timers configured on the switch in XML format:

```
switch(config)# show lldp timers | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:lldp">
  <nf:data>
    <show>
      <lldp>
        <timers>
          <__XML__OPT_Cmd_lldp_show_timers__readonly__>
            <__readonly__>
              <ttl>120</ttl>
              <reinit>2</reinit>
              <tx_interval>30</tx_interval>
              <tx_delay>2</tx_delay>
              <hold_mplier>4</hold_mplier>
              <notification_interval>5</notification_interval>
            </__readonly__>
          </__XML__OPT_Cmd_lldp_show_timers__readonly__>
        </timers>
      </lldp>
    </show>
  </nf:data>
</nf:rpc-reply>
]]]]>
switch(config)#
```