NX-API

This chapter contains the following sections:

About NX-API

In Cisco Nexus devices, CLIs are run only on the device. NX-API improves the accessibility of these CLIs by making them available outside the switch by using HTTP or HTTPS. You can use either of these extensions to the existing Cisco Nexus CLI system on the Cisco Nexus 7000 Series devices. NX-API supports **show** commands and configurations.

NX-API supports JSON-RPC, JSON, and XML formats.

NX-API generates a new certificate on each device when it communicates with them. This new/auto-generated certificate is valid only for 24 hours. NX-API does not use a default or hard-coded or an outdated certificate for its communication.

NX-API comes up with a default self-signed certificate and as mentioned earlier it is valid only for 24 hours. If you need to auto renew the certificate you must re-enable the **feature nxapi** command. You need to use your own certficate without depending on the NX-API certificate. Some browsers might treat this certificate as invalid. In that case you need to add an exception for the same in your settings and continue with your tasks.

Transport

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

The NX-API backend uses the Nginx HTTP server.

Message Format

NX-API is an enhancement to the Cisco Nexus 7000 Series CLI system, which supports XML output.

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 HTTP and HTTPS. If you use HTTPs, all communication to the device is encrypted.

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

Note We recommend that you consider using HTTPS to secure your users' login credentials.

You can enable NX-API by using the **feature manager** command.

Prior to Cisco NX-OS Release 8.2(3) NX-API was accessible on all Layer 3 interfaces and there was no way to restrict the access to a particular VRF. The **nxapi use-vrf** feature is introduced in Cisco NX-OS Release 8.2(3), which helps to secure NX-API by binding the NX-API to a particular VRF.

You can Use ACLs to restrict HTTP or HTTPS access to a device along with VRF, if you want to restrict Nx-API access particular ip. For information about configuring ACLs, see the Cisco Nexus 7000 Series NX-OS Security Configuration Guide.

NX-API provides a session-based cookie, nxapi_auth, when users successfully authenticate for the first time. Along 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 the task of performing the entire 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 multiple authentications helps reduce the devices' workload.

Note

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

Note For information about NX-API response codes, see NX-API Response Codes.

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

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

1. Enable the management interface:

```
switch# configure terminal
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
```

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

```
switch# configure terminal
switch(config)# feature nxapi
```

Example

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

Request

```
<?xml version="1.0"?>
<ins_api>
  <version>1.0</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>show switchname</input>
  <output_format>xml</output_format>
</ins_api>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<ins_api>
  <type>cli_show</type>
  <version>1.0</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": "1.0",
    "type": "cli_show",
    "chunk": "0",
    "sid": "1",
    "input": "show switchname",
```

```
    "output_format": "json"
  }
}
```

Response

```
{
  "ins_api": {
    "type": "cli_show",
    "version": "1.0",
    "sid": "eoc",
    "outputs": {
      "output": {
        "input": "show switchname",
        "msg": "Success",
        "code": "200",
        "body": {
          "hostname": "switch"
        }
      }
    }
  }
}
```

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

Request

```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show switchname",
      "version": 1
    },
    "id": 1
  }
]
```

Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "body": {
      "hostname": "switch"
    }
  },
  "id": 1
}
```

Sending Requests

To send an NX-API request via HTTP, use *http://<ip-address-of-switch>/ins*.

To send an NX-API request with additional security via HTTPS, use *https://<ip-address-of-switch>/ins*.

The IP address of the management interface is *ip-address-of-switch>*.

Note The HTTP request must contain the content-type field in the header. For JSON-RPC requests, this must be application/json-rpc. For proprietary formats, this should be either text/xml or text/json, depending on the input format being used.

Obtaining XSD Files

Step 1 From your browser, navigate to the Cisco software download site:

http://software.cisco.com/download/navigator.html

The Download Software window is displayed.

Step 2 In the Select a Product list, choose **Switches > Data Center Switches >** *platform* > *model* .

Step 3 If you are not already logged in as a registered Cisco user, you are prompted to log in now.

Step 4 From the Select a Software Type list, choose **NX-OS XML Schema Definition**.

Step 5 Find the desired release and click **Download.**

Step 6 If you are requested, follow the instructions to apply for eligibility to download strong encryption software images.

The Cisco End User License Agreement opens.

Step 7 Click **Agree** and follow the instructions to download the file to your PC.

NX-API Sandbox

The NX-API sandbox is a web-based user interface you use to enter the commands, command type, and output type for the Cisco Nexus 7000 Series device using HTTP or HTTPS. After posting the request, the output response is displayed.

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

Use a browser to access the NX-API sandbox.

To view the command reference (that is, description of keywords) from the NX-API Web Interface, click the **Command Reference** link.

Note When using the NX-API sandbox, we recommend that you use the Firefox browser, Release 24.0 or later.

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

- Enable the management interface:

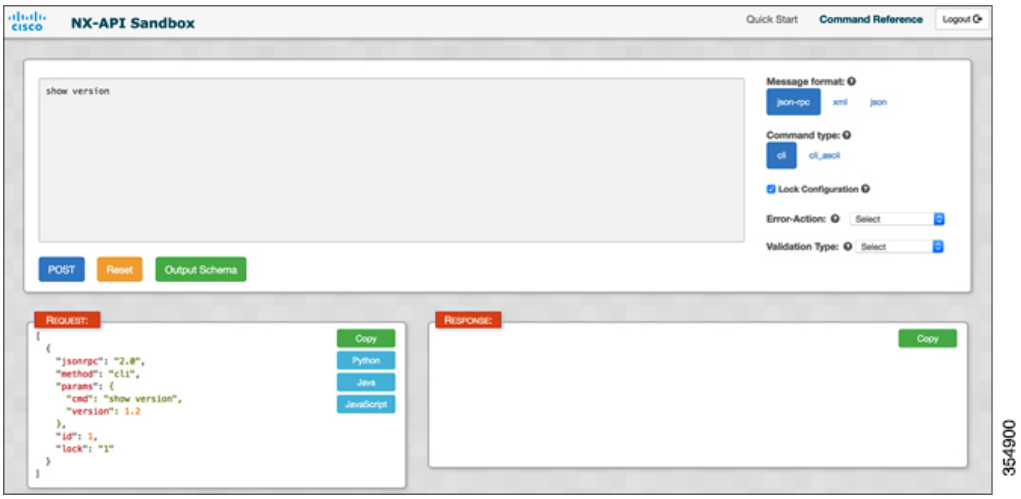
```
switch# configure terminal
switch(config)# interface mgmt 0
switch(config)# ip address 198.51.100.1/24
switch(config)# vrf context managment
switch(config)# ip route 203.0.113.1/0 1.2.3.1
```

- Enable the nxapi feature:

```
switch# configure terminal
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, specify the commands, command type, and output type in the top pane. Click **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.

You can generate Java or JavaScript for each of the request posted through sandbox. To generate Java or Javascript code for each of the requests, click the Java or JavaScript button in the Request pane.

The following sections describe the commands that are available to manage NX-API, and descriptions of the elements in the request and the output responses.

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
feature nxapi	Enables NX-API.
no feature nxapi	Disables NX-API.
nxapi {**http** | **https**} **port** *port*	Specifies a port.
no nxapi {**http** | **https**}	Disables HTTP or HTTPS.

NX-API Management Command	Description
show nxapi	Displays port information, NX-API enable or disable status, and sandbox enable or disable status.
nxapi sandbox	Enables NX-API sandbox.
no nxapi sandbox	Disables NX-API sandbox.
nxapi use-vrf *vrf-name*	Specifies the default VRF, management VRF, or named VRF.
nxapi certificate *certpath* **key** *keypath*	Specifies the upload of the following: • HTTPS certificate when *certpath* is specified. • HTTPS key when *keypath* is specified.

NX-API Request Elements

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

Note A lock will be released by the system if the session that holds the lock is terminated. The session that acquired the lock can perform only necessary configurations.

When the input request format is XML or JSON, use the NX-API elements that are listed in the following table to specify a CLI command:

Table 2: NX-API Request Elements for XML or JSON Format

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

NX-API Request Element	Description
type	Specifies the type of command to be executed. The following command types are supported: • **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. **Note** • Each command is executable only 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
chunk	Some **show** commands can return a large amount of output. For the NX-API client to start processing the output before the entire command is executed, NX-API supports output chunking for **show** commands. Enable or disable chunking with the following settings: **Note** • 0−Do not chunk output. • 1−Chunk output. **Note** • Only **show** commands support chunking. When a series of **show** commands are entered, only the first command is chunked and returned. • The output message format is XML. (This is the default.) Special characters, such as < or >, are converted to form a valid XML message (< is converted into < > is converted into &gt). • You can use XML SAX to parse the chunked output. • When chunking is enabled, the message format is limited to XML. The JSON output format is not supported when chunking is enabled.
roll_back	Specifies the configuration roll-back options. Specify one of the following options. • Stop-on-error—Stops at the first CLI that fails. • Continue-on-error—Ignores and continues with other CLIs. • Rollback-on-error—Performs a rollback to the previous state the system configuration was in. **Note** The roll_back element is available in the cli_conf mode when the input request format is XML or JSON.

NX-API Request Element	Description
validate	Specifies the configuration validation settings. This element allows you to validate the commands before you apply them on the switch. This enables you to verify the consistency of a configuration, for example, the availability of necessary hardware resources, before applying it. Choose the validation type from the Validation Type drop-down list. • **Validate-Only**—Validates the configurations, but does not apply the configurations. • **Validate-and-Set**—Validates the configurations, and applies the configurations on the switch if validation is successful. **Note** The validate element is available in cli_conf mode when the input request format is XML or JSON.
lock	Allows you to specify an exclusive lock on the configuration whereby no other management or programming agent will be able to modify the configuration if this lock is held. **Note** The lock element is available in cli_conf mode when the input request format is XML or JSON.
sid	Specifies the session ID. This element is valid only when the response message is chunked. To retrieve the next chunk of the message, you must specify a *sid* to match the *sid* of the previous response message.
input	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. **Note** Multiple commands are separated with a semi colon (;). The semi colon must be surrounded with single blank characters.) The following are examples of multiple commands: • cli_show–**show version**, **show interface brief**, **show vlan** • cli_conf–**interface Eth4/1**, **no shut**, **switchport**

NX-API Request Element	Description
output_format	The available output message formats are: **Note** • xml−Specifies output in XML format. • json−Specifies output in JSON format. **Note** The Cisco Nexus 7000 Series CLI supports XML output, which means that the JSON output is converted from XML. The conversion is processed on the switch. 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. The content-type header in the HTTP/HTTPS headers indicate the type of response format (XML or JSON).

When JSON-RPC is the input request format, use the NX-API elements that are listed in the following table to specify a CLI command:

Table 3: NX-API Request Elements for JSON-RPC Format

NX-API Request Element	Description
jsonrpc	A string specifying the version of the JSON-RPC protocol. Version must be 2.0.
method	A string containing the name of the method to be invoked. NX-API supports either: • **cli**−show or configuration commands • **cli_ascii**−show or configuration commands; output without formatting
params	A structured value that holds the parameter values used during the invocation of a method. It must contain the following: • **cmd**−CLI command • **version**−NX-API request version identifier

NX-API Request Element	Description
roll_back	Configuration roll-back options. You can specify one of the following options. • Stop-on-error—Stops at the first CLI that fails. • Continue-on-error—Ignores the failed CLI and continues with other CLIs. • Rollback-on-error—Performs a rollback to the previous state the system configuration was in.
validate	Configuration validation settings. This element allows you to validate the commands before you apply them on the switch. This enables you to verify the consistency of a configuration (for example, the availability of necessary hardware resources) before applying it. Choose the validation type from the Validation Type drop-down list. • **Validate-Only**—Validates the configurations, but does not apply the configurations. • **Validate-and-Set**—Validates the configurations, and applies the configurations on the switch if the validation is successful.
lock	An exclusive lock on the configuration can be specified, whereby no other management or programming agent will be able to modify the configuration if this lock is held.
id	An optional identifier established by the client that must contain a string, number, or null value, if it is specified. The value should not be null and numbers contain no fractional parts. If a user does not specify the *id* parameter, the server assumes that the request is simply a notification, resulting in a no response, for example, *id* : 1

NX-API Response Elements

NX-API Response Elements for XML or JSON Requests

When the input request is in XML or JSON format, the response contains the following elements:

Table 4: NX-API Response Elements

NX-API Response Element	Description
version	NX-API version.

NX-API Response Element	Description
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	Tag that encloses all command outputs. When multiple commands are in cli_show or cli_show_ascii, each command output is enclosed by a single output tag. When the message type is cli_conf, there is a single output tag for all the commands because cli_conf commands require context.
output	Tag that encloses the output of a single command output. For cli_conf message type, this element contains the outputs of all the commands.
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 at http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml.
msg	Error message associated with the returned error code.

NX-API Response Elements for JSON-RPC Requests

The response object of all JSON-RPC requests will be in JSON-RPC 2.0 response format as defined in the following table.

NX-API Response Element	Description
jsonrpc	A string specifying the version of the JSON-RPC protocol. The version is 2.0.
result	This field is included only upon receiving a successful response. The value of this field contains the requested CLI output.

NX-API Response Element	Description
error	This field is included only on error (mutually exclusive with the result object). The error object contains the following: • code — An integer error code as specified by the JSON-RPC specification. • message — A human-readable string to correspond with the error code. • data — An optional structure that contains useful information (such as CLI error messages or additional information).
id	The same value as the **id** in the corresponding request object. When there is a problem parsing the **id** in the request, this value is null.

Additional References

This section provides a list of sections that provide additional information about implementing NX-API:

- NX-API DevNet Community
- NX-API Github (Nexus 7000)
- NX-API Github (NX-OS Programmability scripts)