



## NX-API Developer Sandbox

- [NX-API Developer Sandbox: NX-OS Releases Prior to 9.2\(2\)](#), on page 1

## NX-API Developer Sandbox: NX-OS Releases Prior to 9.2(2)

### About the NX-API Developer Sandbox

The NX-API Developer Sandbox is a web form hosted on the switch. It translates NX-OS CLI commands into equivalent XML or JSON payloads.

The web form is a single screen with three panes — Command (top pane), Request, and Response — as shown in the figure.

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

The screenshot shows the NX-API Developer Sandbox web interface. At the top, there's a Cisco logo and the title 'NX-API Developer Sandbox'. Below this, there's a 'Quick Start' link and a 'Logout' button. The main area is divided into three panes. The top pane is the 'Command' pane, which contains a text input field with 'show version', a 'POST' button, and a 'Reset' button. To the right of the Command pane are two dropdown menus: 'Message format' (set to 'json-rpc') and 'Command type' (set to 'cli'). The bottom pane is split into two sections: 'REQUEST' and 'RESPONSE'. The 'REQUEST' section shows a JSON payload: 

```
{  "jsonrpc": "2.0",  "method": "cli_ascii",  "params": {    "cmd": "show version",    "version": 1  },  "id": 1}
```

. The 'RESPONSE' section shows the corresponding JSON response: 

```
{  "jsonrpc": "2.0",  "result": {    "msg": "Cisco Nexus Operating System (NX-OS) Software\nTAC support: http://www.cisco.com/tac\nDocuments: http://www.cisco.com/dteng/CTE/DKHS/1.1/CTE_DKHS_1.1.html"  },  "id": 1}
```

. The interface also includes a 'Copy' button for the request and response JSON.

Controls in the Command pane allow you to choose a message format for a supported API, such as NX-API REST, and a command type, such as XML or JSON. The available command type options vary depending on the selected message format.

When you type or paste one or more CLI commands into the Command pane, the web form converts the commands into an API payload, checking for configuration errors, and displays the resulting payload in the Request pane. If you then choose to post the payload directly from the Sandbox to the switch, using the POST button in the Command pane, the Response pane displays the API response.

## Guidelines and Limitations

Following are the guidelines and limitations for the Developer Sandbox:

- Clicking **POST** in the Sandbox commits the command to the switch, which can result in a configuration or state change.
- Some feature configuration commands are not available until their associated feature has been enabled.

## Configuring the Message Format and Command Type

The **Message Format** and **Command Type** are configured in the upper right corner of the Command pane (the top pane). For **Message Format**, choose the format of the API protocol that you want to use. The Developer Sandbox supports the following API protocols:

**Table 1: NX-OS API Protocols**

Protocol	Description
json-rpc	A standard lightweight remote procedure call (RPC) protocol that can be used to deliver NX-OS CLI commands in a JSON payload. The JSON-RPC 2.0 specification is outlined by <a href="http://jsonrpc.org">jsonrpc.org</a> .
xml	Cisco NX-API proprietary protocol for delivering NX-OS CLI or bash commands in an XML payload.
json	Cisco NX-API proprietary protocol for delivering NX-OS CLI or bash commands in a JSON payload.
nx-api rest	Cisco NX-API proprietary protocol for manipulating and reading managed objects (MOs) and their properties in the internal NX-OS data management engine (DME) model. For more information, see the <a href="#">Cisco Nexus NX-API References</a> .
nx yang	The YANG ("Yet Another Next Generation") data modeling language for configuration and state data.

When the **Message Format** has been chosen, a set of **Command Type** options are presented just below the **Message Format** control. The **Command Type** setting can constrain the input CLI and can determine the **Request** and **Response** format. The options vary depending on the **Message Format** selection. For each **Message Format**, the following table describes the **Command Type** options:

Table 2: Command Types

Message format	Command type
json-rpc	<ul style="list-style-type: none"> <li>cli — show or configuration commands</li> <li>cli-ascii — show or configuration commands, output without formatting</li> </ul>
xml	<ul style="list-style-type: none"> <li>cli_show — show commands. If the command does not support XML output, an error message will be returned.</li> <li>cli_show_ascii — show commands, output without formatting</li> <li>cli_conf — configuration commands. Interactive configuration commands are not supported.</li> <li>bash — bash commands. Most non-interactive bash commands are supported.</li> </ul> <p><b>Note</b> The bash shell must be enabled in the switch.</p>
json	<ul style="list-style-type: none"> <li>cli_show — show commands. If the command does not support XML output, an error message will be returned.</li> <li>cli_show_ascii — show commands, output without formatting</li> <li>cli_conf — configuration commands. Interactive configuration commands are not supported.</li> <li>bash — bash commands. Most non-interactive bash commands are supported.</li> </ul> <p><b>Note</b> The bash shell must be enabled in the switch.</p>
nx-api rest	<ul style="list-style-type: none"> <li>cli — configuration commands</li> </ul>
nx yang	<ul style="list-style-type: none"> <li>json — JSON structure is used for payload</li> <li>xml — XML structure is used for payload</li> </ul>

### Output Chunking

In order to handle large show command output, some NX-API message formats support output chunking for show commands. In this case, an **Enable chunk mode** checkbox appears below the **Command Type** control along with a session ID (**SID**) type-in box.

When chunking is enabled, the response is sent in multiple "chunks," with the first chunk sent in the immediate command response. In order to retrieve the next chunk of the response message, you must send an NX-API request with **SID** set to the session ID of the previous response message.

## Using the Developer Sandbox

### Using the Developer Sandbox to Convert CLI Commands to Payloads



---

**Tip** Online help is available by clicking **Quick Start** in the upper right corner of the Sandbox window. Additional details, such as response codes and security methods, can be found in the NX-API CLI chapter. Only configuration commands are supported.

---

#### Procedure

---

- Step 1** Configure the **Message Format** and **Command Type** for the API protocol you want to use. For detailed instructions, see [Configuring the Message Format and Command Type, on page 2](#).
- Step 2** Type or paste NX-OS CLI configuration commands, one command per line, into the text entry box in the top pane.
- You can erase the contents of the text entry box (and the **Request** and **Response** panes) by clicking **Reset** at the bottom of the top pane.

Enter CLI commands here, one command per line.

Message format:   
 json-rpc   xml   json  
 nx-api rest   nx yang

Command type:   
 cli   model

Convert   Reset

CLI   Copy

ERROR   Copy

Copyright © 2014-2016 Cisco Systems, Inc. All rights reserved.   NX-API version 1.1

**Step 3** Click the **Convert** at the bottom of the top pane.

If the CLI commands contain no configuration errors, the payload appears in the **Request** pane. If errors are present, a descriptive error message appears in the **Response** pane.

The screenshot shows the NX-API Developer Sandbox interface. At the top, there is a header with the Cisco logo, the title "NX-API Developer Sandbox", and links for "Quick Start" and "Logout". The main area is divided into two panes: "Request" and "Response".

In the "Request" pane, there is a text area containing a JSON payload:

```
api/mo/sys.json
{
  "topSystem": {
    "attributes": {
      "name": "REST2CLI"
    }
  }
}
```

Below the text area are "Convert" and "Reset" buttons. To the right of the text area, there are dropdown menus for "Message format" (with options: json-rpc, xml, json) and "Command type" (with options: cli, model). The "nx-api rest" and "nx yang" buttons are also visible.

In the "Response" pane, there is a red header labeled "ERROR:" and a "Copy" button. The response area is currently empty.

At the bottom of the interface, there is a status bar with the text "Waiting for bam.nr-data.net..." and a copyright notice: "Copyright © 2014-2016 Cisco Systems, Inc. All rights reserved." The version "NX-API version 1.1" is also displayed.

**Step 4**

When a valid payload is present in the **Request** pane, you can click **POST** to send the payload as an API call to the switch.

The response from the switch appears in the **Response** pane.

**Warning** Clicking **POST** commits the command to the switch, which can result in a configuration or state change.

The screenshot shows the NX-API Developer Sandbox interface. At the top, there's a header with the Cisco logo, the title "NX-API Developer Sandbox", and links for "Quick Start" and "Logout". Below the header is a large text area containing the CLI command "logging level netstack 6". To the right of this text area are two sections: "Message format:" with buttons for "json-rpc", "xml", "json", "nx-api rest" (which is highlighted), and "nx yang"; and "Command type:" with buttons for "cli" (highlighted) and "model". Below the text area are three buttons: "POST", "Reset", and "Convert". At the bottom, there are two panels: "REQUEST:" and "RESPONSE:". The "REQUEST:" panel shows a JSON payload with a nested structure for "topSystem", "children", "ipv4Entity", "children", "ipv4Inst", and "attributes", including a "loggingLevel" of "informational". To the right of the JSON are buttons for "Copy", "Python", and "Python3". The "RESPONSE:" panel shows a simple JSON object with "imdata": [] and a "Copy" button.

```
logging level netstack 6
```

Message format:   
 json-rpc xml json   
 nx-api rest nx yang

Command type:   
 cli model

POST Reset Convert

**REQUEST:**

```
{  
  "topSystem": {  
    "children": [  
      {  
        "ipv4Entity": {  
          "children": [  
            {  
              "ipv4Inst": {  
                "attributes": {  
                  "loggingLevel": "informational"  
                }  
              }  
            }  
          ]  
        }  
      }  
    ]  
  }  
}
```

Copy Python Python3

**RESPONSE:**

```
{  
  "imdata": []  
}
```

Copy

- Step 5** You can copy the contents of the **Request** or **Response** pane to the clipboard by clicking **Copy** in the pane.
- Step 6** You can obtain a Python implementation of the request on the clipboard by clicking **Python** in the **Request** pane.

