



NX-API Developer Sandbox

This chapter contains the following sections:

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

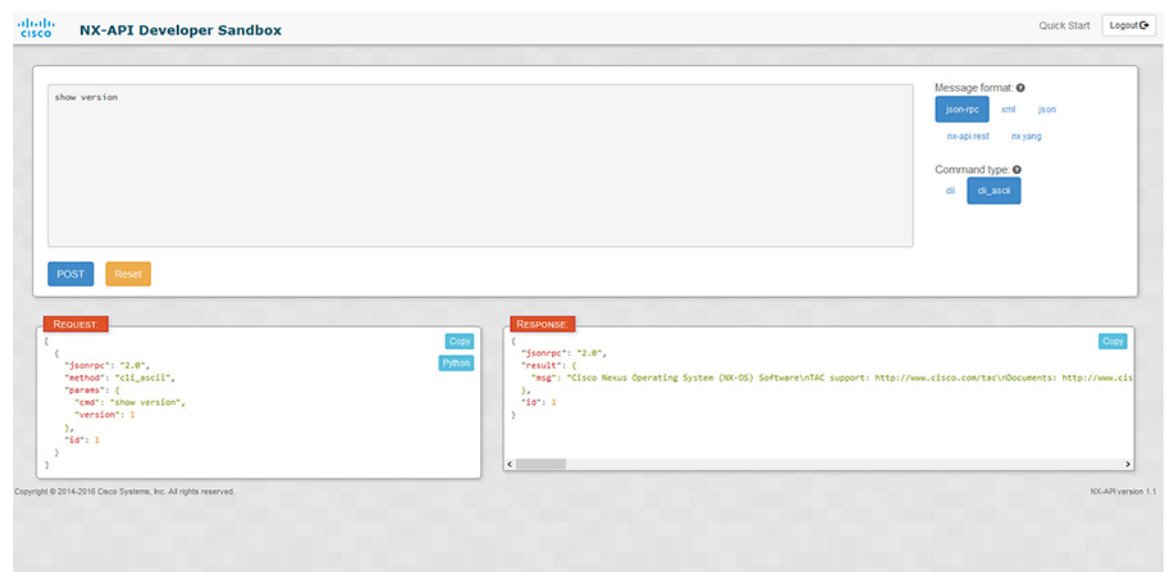
NX-API Developer Sandbox: NX-OS Releases Prior to 9.2(2)

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



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

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



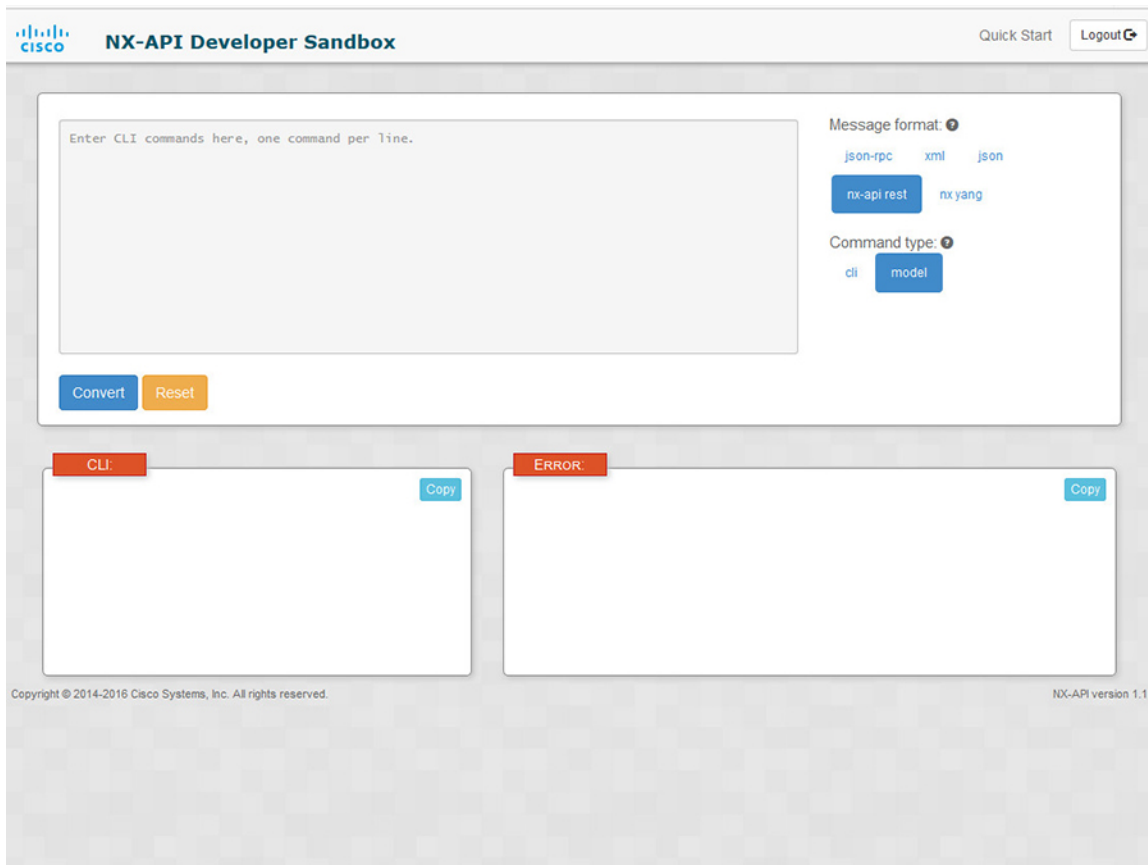
Tip Online help is available by clicking **Quick Start** in the upper right corner of the Sandbox window. Additional details, such as response codes and security methods, can be found in the NX-API CLI chapter. Only configuration commands are supported.

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.



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 displays the NX-API Developer Sandbox interface. At the top left is the Cisco logo and the title "NX-API Developer Sandbox". On the top right are "Quick Start" and "Logout" links. The main area is divided into several sections:

- Request Pane:** Contains a text area with a JSON payload:

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

Below the text area are "Convert" and "Reset" buttons. To the right, 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 "model" options are selected.
- CLI Pane:** Labeled "CLI:" in a red header, it contains the text "hostname REST2CLI" and a "Copy" button.
- ERROR Pane:** Labeled "ERROR:" in a red header, it is currently empty and has a "Copy" button.

At the bottom left, there is a status bar that says "Waiting for bam.nr-data.net...". At the bottom right, there is a copyright notice: "Copyright © 2014-2016 Cisco Systems, Inc. All rights reserved." and the version "NX-API version 1.1".

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 left is the Cisco logo and the title "NX-API Developer Sandbox". At the top right are "Quick Start" and "Logout" links. The main area contains a text input field with the command "Logging level netstack 6". To the right of the input field are two sections: "Message format:" with options "json-rpc", "xml", and "json" (selected), and "Command type:" with options "cli" (selected) and "model". Below the input field are three buttons: "POST" (orange), "Reset" (yellow), and "Convert" (blue). Below the input field are two panels: "REQUEST:" and "RESPONSE:". The "REQUEST:" panel shows a JSON payload with a "loggingLevel" of "informational" and has buttons for "Copy", "Python", and "Python3". The "RESPONSE:" panel shows a JSON payload with an empty "imdata" array and has a "Copy" button.

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.