# NX-API Developer Sandbox



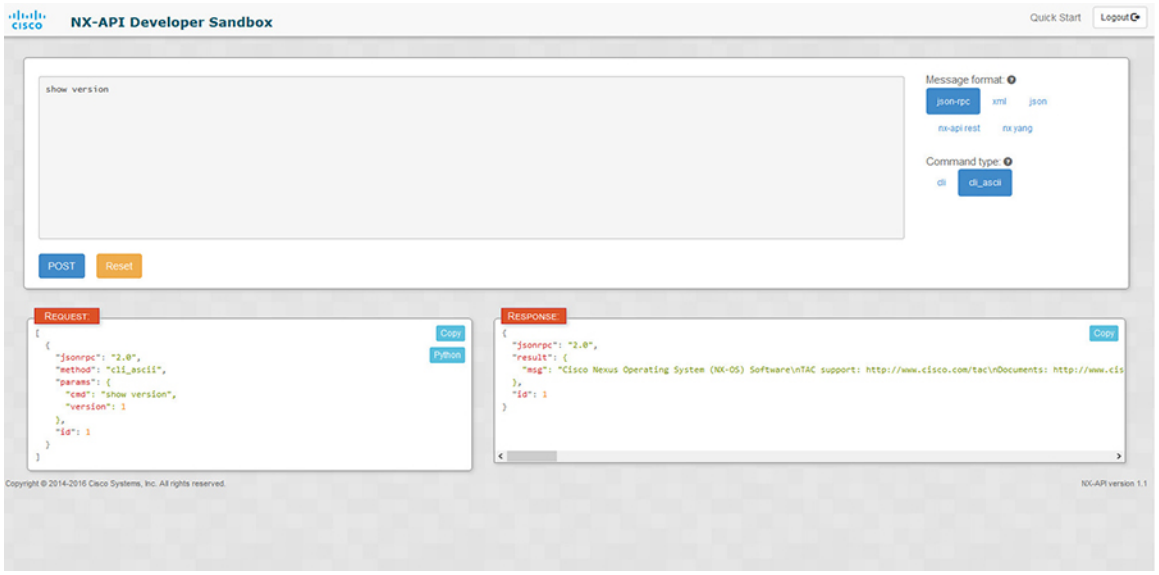
# NX-API Developer Sandbox: NX-OS Releases Prior to 9.2(2)

## 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 **Send** 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. For example, configuring a BGP router requires first enabling BGP with the **feature bgp** command. Similarly, configuring an OSPF router requires first enabling OSPF with the **feature ospf** command. This also applies to **evpn esi multihoming**, which enables its dependent commands such as **evpn multihoming core-tracking**. For more information about enabling features to access feature dependent commands, see the .
- Using Sandbox to convert with DN is supported only for finding the DN of a CLI config. Any other workflow, for example, using DME to convert DN for CLI configuration commands is not supported.
- The Command pane (the top pane) supports a maximum of 10,000 individual lines of input.

# 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 about the Cisco Nexus 3000 and 9000 Series NX-API REST SDK, see https://developer.cisco.com/site/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 | • cli — show or configuration commands<br>• cli-ascii — show or configuration commands, output without formatting |
| xml | • cli_show — show commands. If the command does not support XML output, an error message will be returned.<br>• cli_show_ascii — show commands, output without formatting<br>• cli_conf — configuration commands. Interactive configuration commands are not supported.<br>• bash — bash commands. Most non-interactive bash commands are supported.<br>**Note** The bash shell must be enabled in the switch. |
| json | • cli_show — show commands. If the command does not support XML output, an error message will be returned.<br>• cli_show_ascii — show commands, output without formatting<br>• cli_conf — configuration commands. Interactive configuration commands are not supported.<br>• bash — bash commands. Most non-interactive bash commands are supported.<br>**Note** The bash shell must be enabled in the switch. |
| nx-api rest | • cli — configuration commands |
| nx yang | • json — JSON structure is used for payload<br>• 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 REST 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 chapter "NX-API CLI".

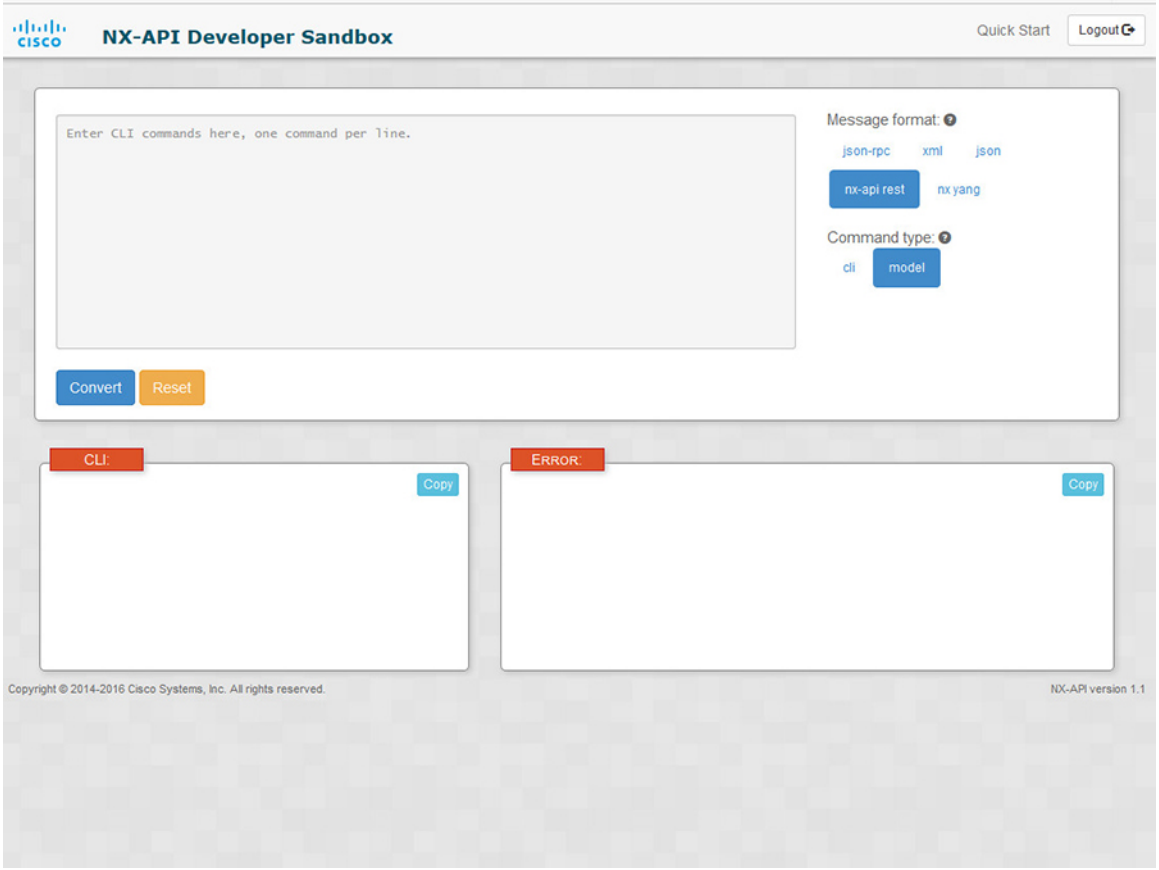
Only configuration commands are supported.

**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.

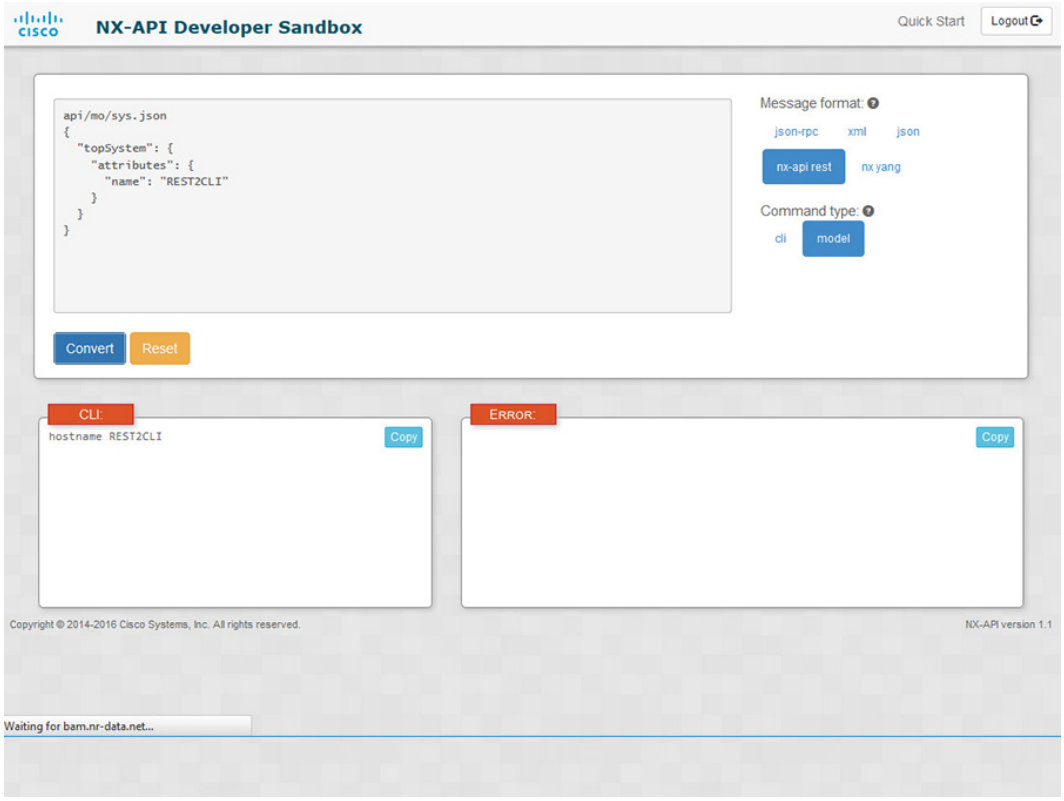
**Step 2** Type or paste NX-OS CLI configuration commands, one command per line, into the text entry box in the top pane.

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.

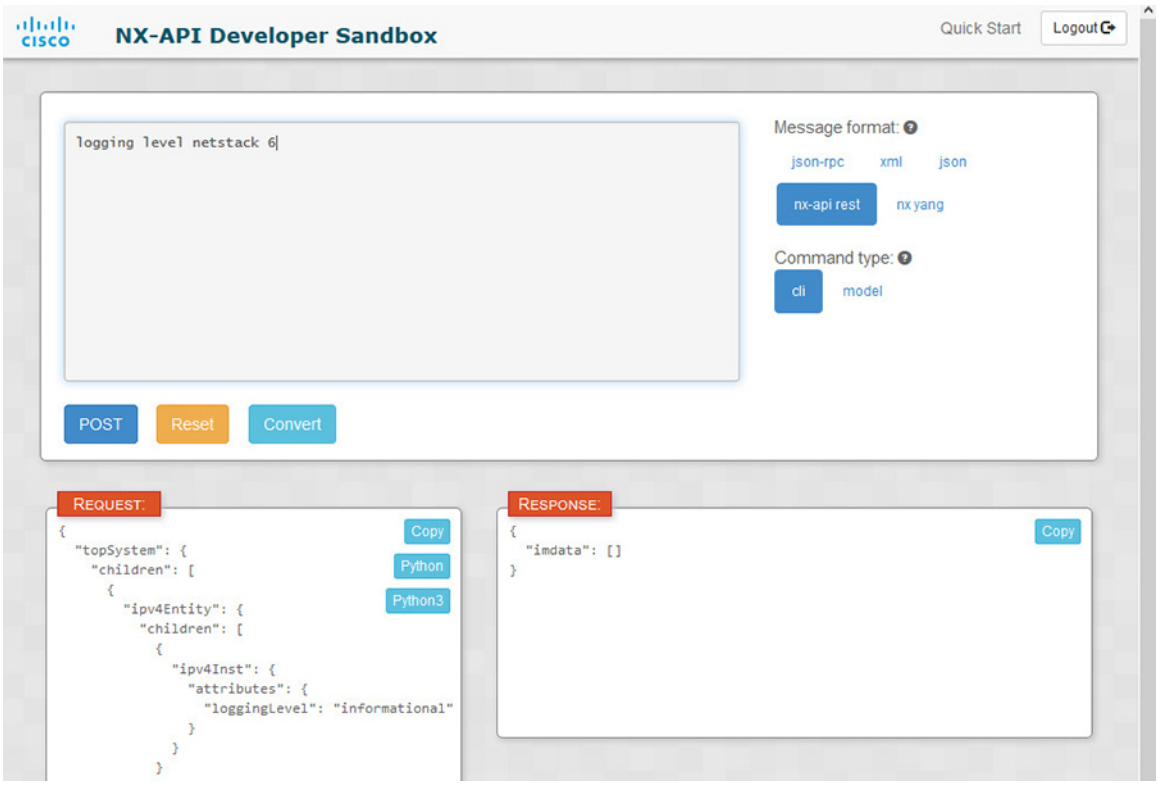
If the CLI commands contain no configuration errors, the payload appears in the **Request** pane. If errors are present, a descriptive error message appears in the **Response** pane.




**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.




**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.

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

**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 chapter "NX-API CLI".

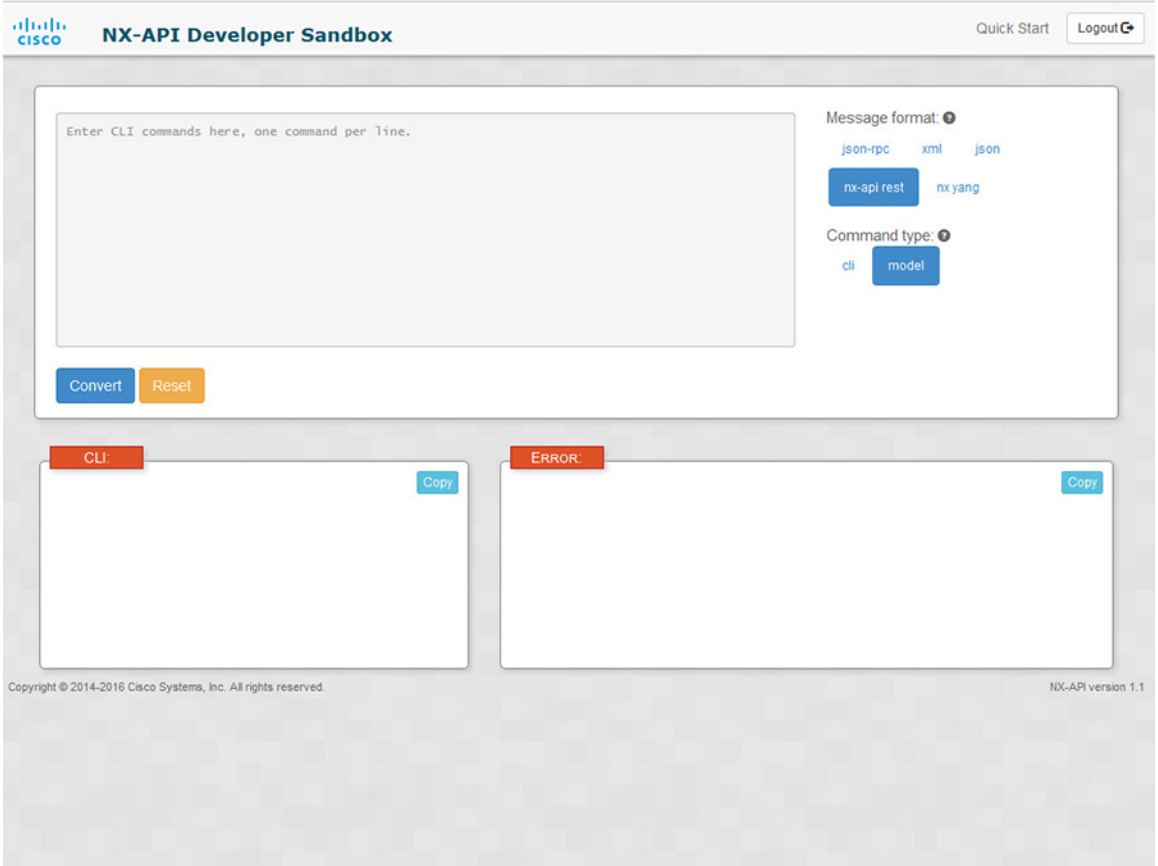
### SUMMARY STEPS

1. Select **`nx-api rest`** as the **Message Format** and **`model`** as the **Command Type**.
2. Enter a DN and payload into the text entry box in the top pane. Then click on the **Convert** button below the top pane.

### DETAILED STEPS

**Step 1** Select **`nx-api rest`** as the **Message Format** and **`model`** as the **Command Type**.

**Example:**

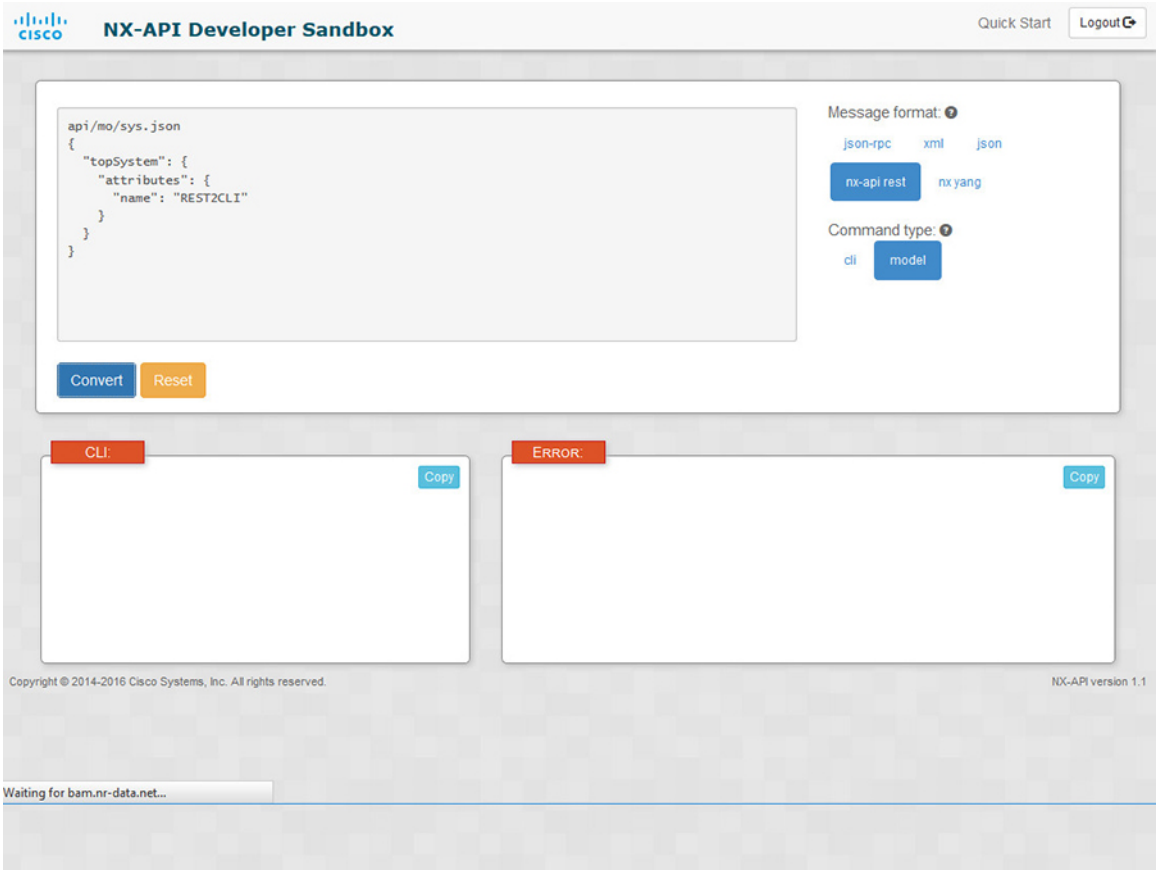



**Step 2** Enter a DN and payload into the text entry box in the top pane. Then click on the **Convert** button below the top pane.

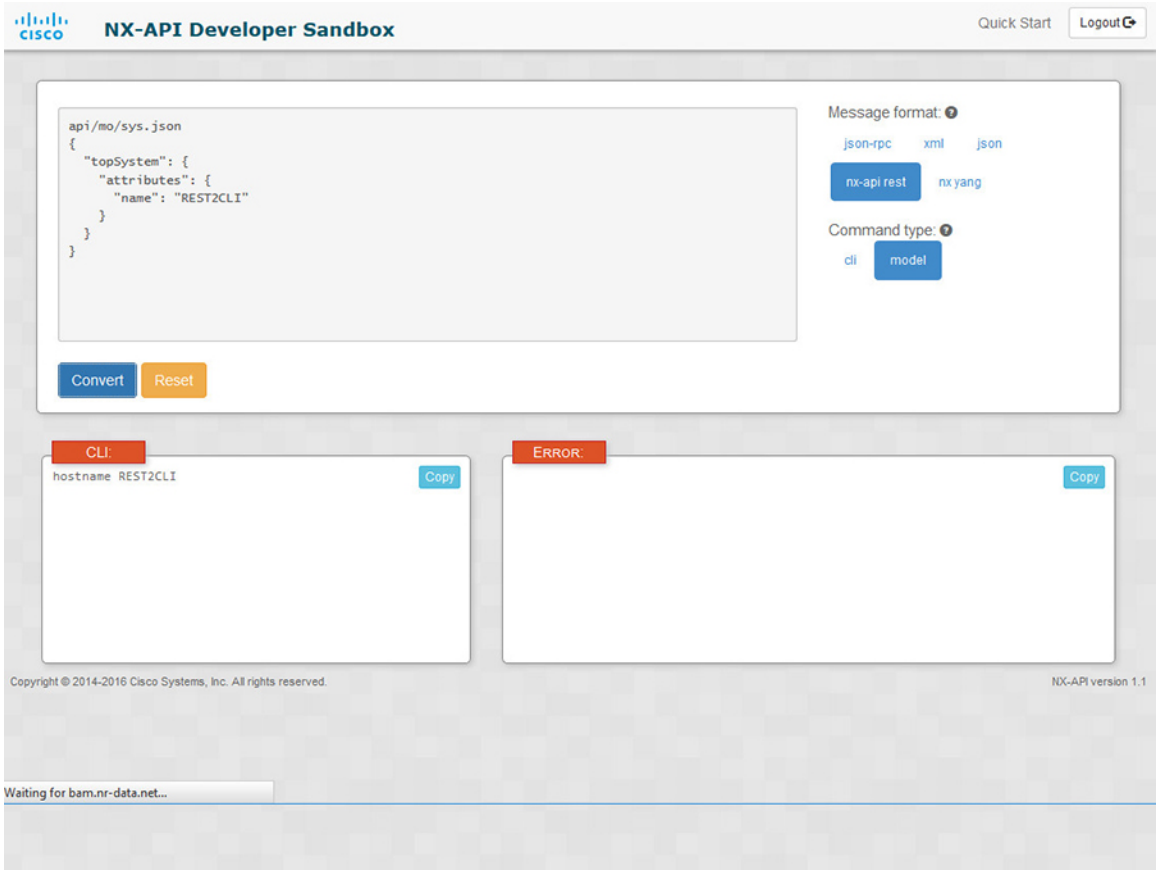
**Example:**

For this example, the DN is **`api/mo/sys.json`** and the NX-API REST payload is:

```
{
  "topSystem": {
    "attributes": {
      "name": "REST2CLI"
    }
  }
}
```




When you click on the **Convert** button, the CLI equivalent appears in the **CLI** pane as shown in the following image.


NX-API Developer Sandbox
Quick Start
Logout
api/mo/sys.json
{
"topSystem": {
"attributes": {
"name": "REST2CLI"
}
}
}
Message format:
json-rpc
xml
json
nx-api rest
nx yang
Command type:
cli
model
Convert
Reset
CLI:
hostname REST2CLI
Copy
ERROR:
Copy
Copyright © 2014-2016 Cisco Systems, Inc. All rights reserved.
NX-API version 1.1
Waiting for bam.nr-data.net...

**Note** The Developer Sandbox cannot convert all payloads into equivalent CLIs, even if the Sandbox converted the CLIs to NX-API REST payloads. The following is a list of possible sources of error that can prevent a payload from completely converting to CLI commands:

***Table 3: Sources of REST2CLI Errors***

| Payload Issue | Result |
|---|---|
| The payload contains an attribute that does not exist in the MO.<br>Example:<br>`api/mo/sys.json`<br>`{ "topSystem": { "children": [ { "interfaceEntity": { "children": [ { "l1PhysIf": { "attributes": { "id": "eth1/1", "fakeattribute": "totallyFake" } } } ] } } ] } }` | The **Error** pane will return an error related to the attribute.<br>Example:<br>**CLI**<br>**Error** `unknown attribute 'fakeattribute' in element 'l1PhysIf'` |
| The payload includes MOs that aren't yet supported for conversion:<br>Example:<br>`api/mo/sys.json`<br>`{ "topSystem": { "children": [ { "dhcpEntity": { "children": [ { "dhcpInst": { "attributes": { "SnoopingEnabled": "yes" } } } ] } } ] } }` | The **Error** Pane will return an error related to the unsupported MO.<br>Example:<br>**CLI**<br>**Error** `The entire subtree of "sys/dhcp" is not converted.` |

# NX-API Developer Sandbox: NX-OS Release 9.2(2) and Later

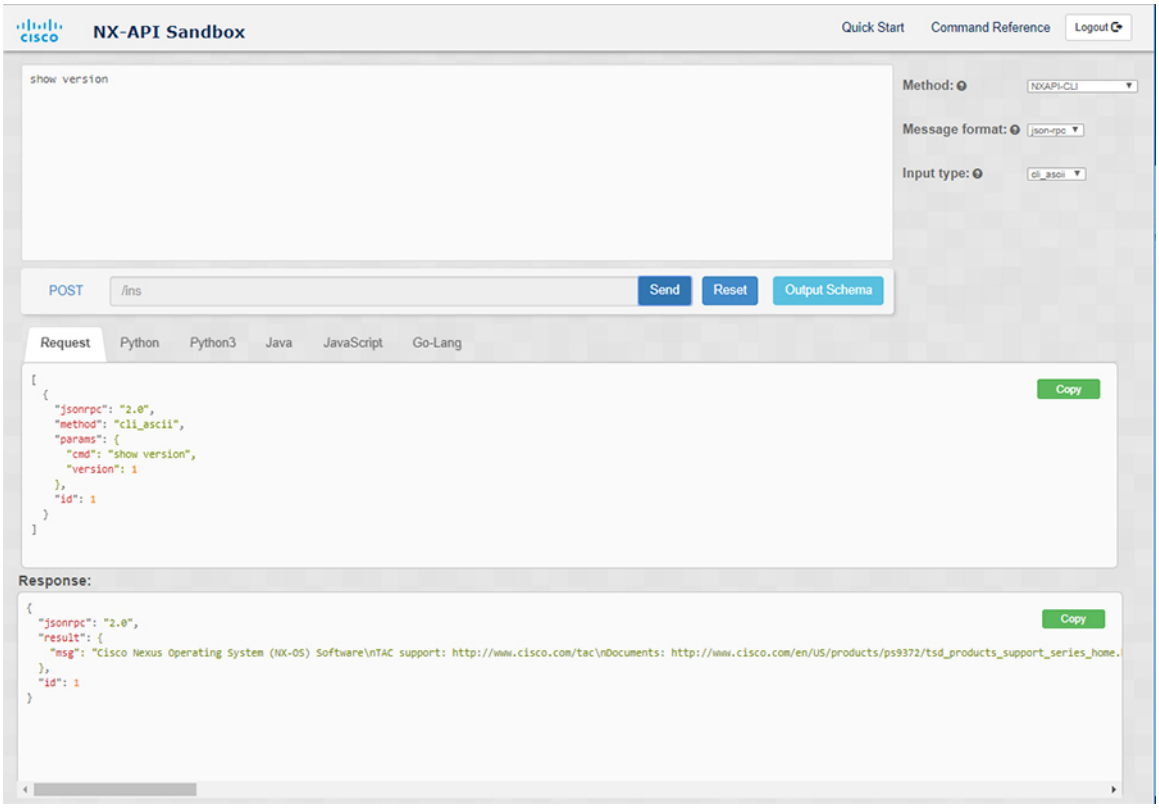
## About the NX-API Developer Sandbox

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

The web form is a single screen with three panes — Command (top pane), Request (middle pane), and Response (bottom pane) — as shown in the figure below. The designated name (DN) field is located between the Command and Request panes (seen in the figure below located between the **POST** and **Send** options).

The Request pane also has a series of tabs. Each tab represents a different language: **Python**, **Python3**, **Java**, **JavaScript**, and **Go-Lang**. Each tab enables you to view the request in the respective language. For example, after converting CLI commands into an XML or JSON payload, click the **Python** tab to view the request in Python, which you can use to create scripts.

***Figure 2: NX-API Developer Sandbox with Example Request and Output Response***




Controls in the Command pane enable you to choose a supported API, such as NX-API REST, an input type, such as model (payload) or CLI, and a message format, such as XML or JSON. The available options vary depending on the chosen method.

When you choose the NXAPI-REST (DME) method, type or paste one or more CLI commands into the Command pane,and click **Convert**, the web form converts the commands into a REST 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 (by choosing the **POST** option and clicking **SEND**), the Response pane displays the API response. For more information, see Using the Developer Sandbox to Convert CLI Commands to REST Payloads, on page 16

Conversely, the Cisco NX-API Developer Sandbox checks the payload for configuration errors then displays the equivalent CLIs in the Response pane. For more information, see Using the Developer Sandbox to Convert from REST Payloads to CLI Commands, on page 19

# Guidelines and Limitations

Following are the guidelines and limitations for the Developer Sandbox:

- Clicking **Send** 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. For example, configuring a BGP router requires first enabling BGP with the **feature bgp** command. Similarly, configuring an OSPF router requires first enabling OSPF with the **feature ospf** command. This also applies to **evpn esi multihoming**, which enables its dependent commands such as **evpn multihoming core-tracking**. For more information about enabling features to access feature dependent commands, see the .
- Using Sandbox to convert with DN is supported only for finding the DN of a CLI config. Any other workflow, for example, using DME to convert DN for CLI configuration commands is not supported.
- The Command pane (the top pane) supports a maximum of 10,000 individual lines of input.

# Configuring the Message Format and Input Type

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

***Table 4: NX-OS API Protocols***

| Protocol | Description |
|---|---|
| NXAPI-CLI | Cisco NX-API proprietary protocol for delivering NX-OS CLI or bash commands in an XML or a JSON payload. |

| Protocol | Description |
|---|---|
| NXAPI-REST (DME) | 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. The NXAPI-REST (DME) protocol displays a drop-down list that enables you to choose from the following methods:<br>• **POST**<br>• **GET**<br>• **PUT**<br>• **DELETE**<br>For more information about the Cisco Nexus 3000 and 9000 Series NX-API REST SDK, see https://developer.cisco.com/site/cisco-nexus-nx-api-references/. |
| RESTCONF (Yang) | The YANG ("Yet Another Next Generation") data modeling language for configuration and state data.<br>The RESTCONF (Yang) protocol displays a drop-down list that enables you to choose from the following methods:<br>• **POST**<br>• **GET**<br>• **PUT**<br>• **PATCH**<br>• **DELETE** |

When you choose the **Method**, a set of **Message format** or **Input type** options are displayed in a drop-down list. The **Message format** can constrain the input CLI and determine the **Request** and **Response** format. The options vary depending on the **Method** you choose.

The following table describes the **Input/Command type** options for each **Message format**:

***Table 5: Command Types***

| Method | Message format | Input/Command type |
|---|---|---|
| NXAPI-CLI | json-rpc | • cli — show or configuration commands<br>• cli-ascii — show or configuration commands, output without formatting<br>• cli-array — show commands. Similar to cli, but with cli_array, data is returned as a list of one element, or an array, within square brackets, [ ]. |

| Method | Message format | Input/Command type |
|---|---|---|
| NXAPI-CLI | xml | • cli_show — show commands. If the command does not support XML output, an error message will be returned.<br>• cli_show_ascii — show commands, output without formatting<br>• cli_conf — configuration commands. Interactive configuration commands are not supported.<br>• bash — bash commands. Most non-interactive bash commands are supported.<br>**Note** The bash shell must be enabled in the switch. |
| NXAPI-CLI | json | • cli_show — show commands. If the command does not support XML output, an error message will be returned.<br>**Note** Beginning with Cisco NX-OS Release 9.3(3), the cli_show_array command is recommended over the cli_show command.<br>• cli_show_array — show commands. Similar to cli_show, but with cli_show_array, data is returned as a list of one element, or an array, within square brackets [ ].<br>• cli_show_ascii — show commands, output without formatting<br>• cli_conf — configuration commands. Interactive configuration commands are not supported.<br>• bash — bash commands. Most non-interactive bash commands are supported.<br>**Note** The bash shell must be enabled in the switch. |
| NXAPI-REST (DME) | | • cli — CLI to model conversion<br>• model — Model to CLI conversion. |
| RESTCONF (Yang) | • json — JSON structure is used for payload<br>• 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** check box 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 REST Payloads

**Tip**

- Online help is available by clicking the help icons (**?**) next to the field names located in the upper-right corner of the Cisco NX-API Developer Sandbox window.
- For additional details, such as response codes and security methods, see the *NX-API CLI* chapter.
- Only configuration commands are supported.

The Cisco NX-API Developer Sandbox enables you to convert CLI commands to REST payloads.

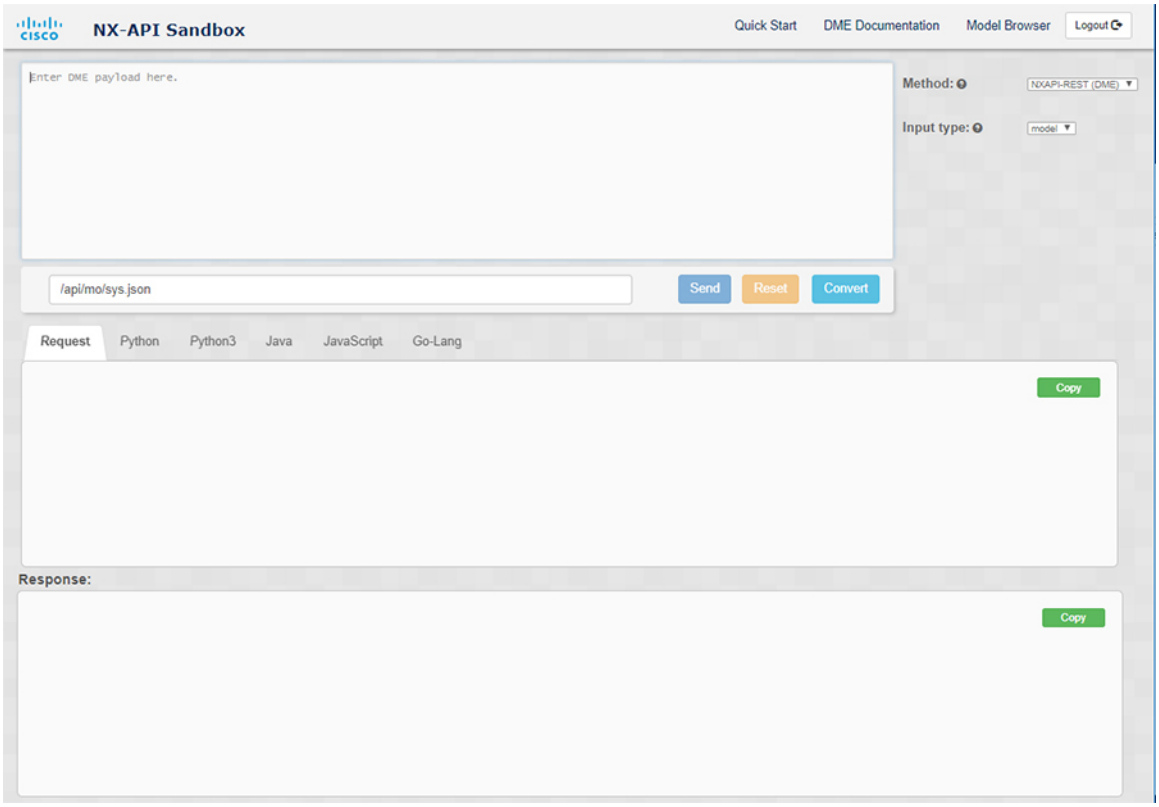
**Step 1** Click the **Method** drop-down list and choose **NXAPI-REST (DME)**.

The **Input** type drop-down list appears.

**Step 2** Click the **Input** type drop-down list and choose **cli**.

**Step 3** 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.



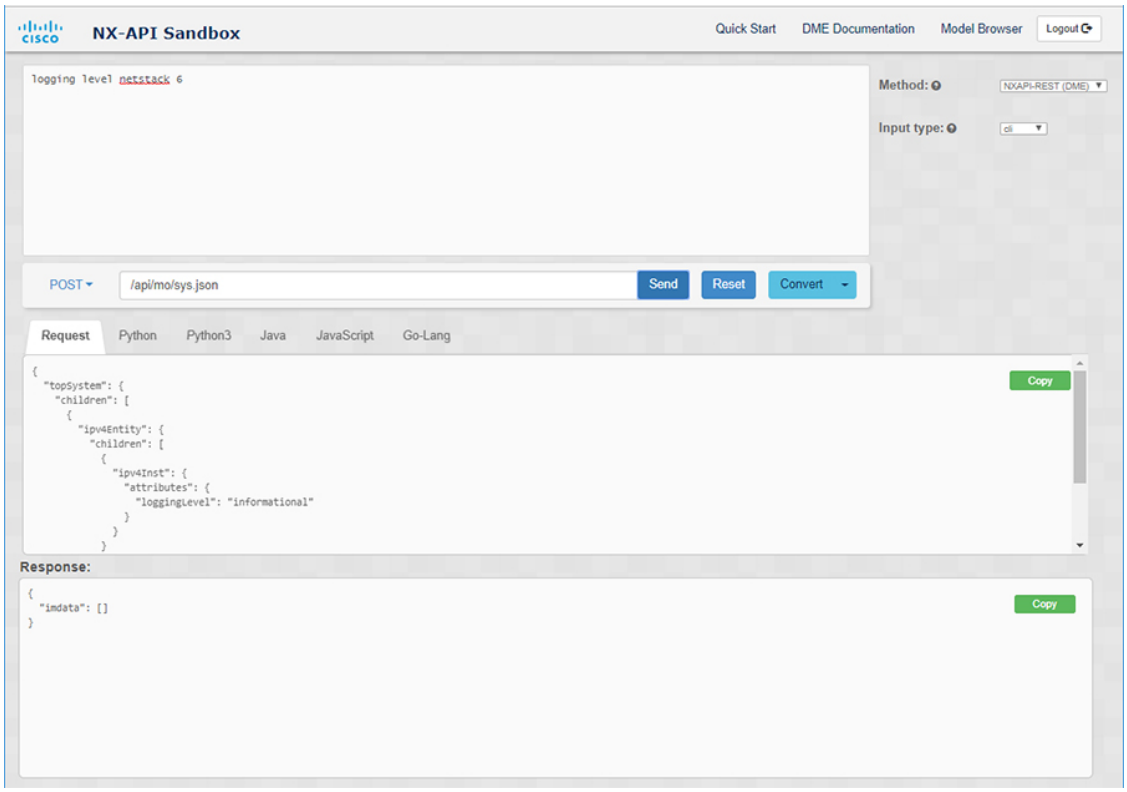

**Step 4** Click **Convert**.

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.

**Step 5** (Optional) To send a valid payload as an API call to the switch, click **Send**.

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

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




**Step 6** (Optional) To obtain the DN for an MO in the payload:

**a.** From the **Request** pane, choose **POST**.

**b.** Click the **Convert** drop-down list and choose **Convert (with DN)**.

The payload appears with with a **dn** field that contains the DN that corresponds to each MO in the payload.

**Step 7** (Optional) To overwrite the current configuration with a new configuration:

**a.** Click the **Convert** drop-down list and choose **Convert (for Replace)**. The **Request** pane displays a payload with a **status** field set to **replace**.

**b.** From the **Request** pane, choose **POST**.

**c.** Click **Send**.

The current configuration is replaced with the posted configuration. For example, if you start with the following configuration:

```
interface eth1/2
  description test
  mtu 1501
```

Then use **Convert (for Replace)** to POST the following configuration:

```
interface eth1/2
  description testForcr
```

The `mtu` configuration is removed and only the new description (`testForcr`) is present under the interface. This change is confirmed when entering **show running-config** .

**Step 8** (Optional) To copy the contents of a pane, such as the **Request** or **Response** pane, click **Copy**. The contents o the respective pane is copied to the clipboard.

**Step 9** (Optional) To convert the request into an of the formats listed below, click on the appropriate tab in the **Request** pane:

- **Python**
- **Python3**
- **Java**
- **JavaScript**
- **Go-Lang**

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

The Cisco NX-API Developer Sandbox enables you to convert REST payloads to corresponding CLI commands. This option is only available for the NXAPI-REST (DME) method.

**Tip**

- Online help is available by clicking help icons (**?**) next to the Cisco NX-API Developer Sandbox field names. Click a help icon get information about the respective field.

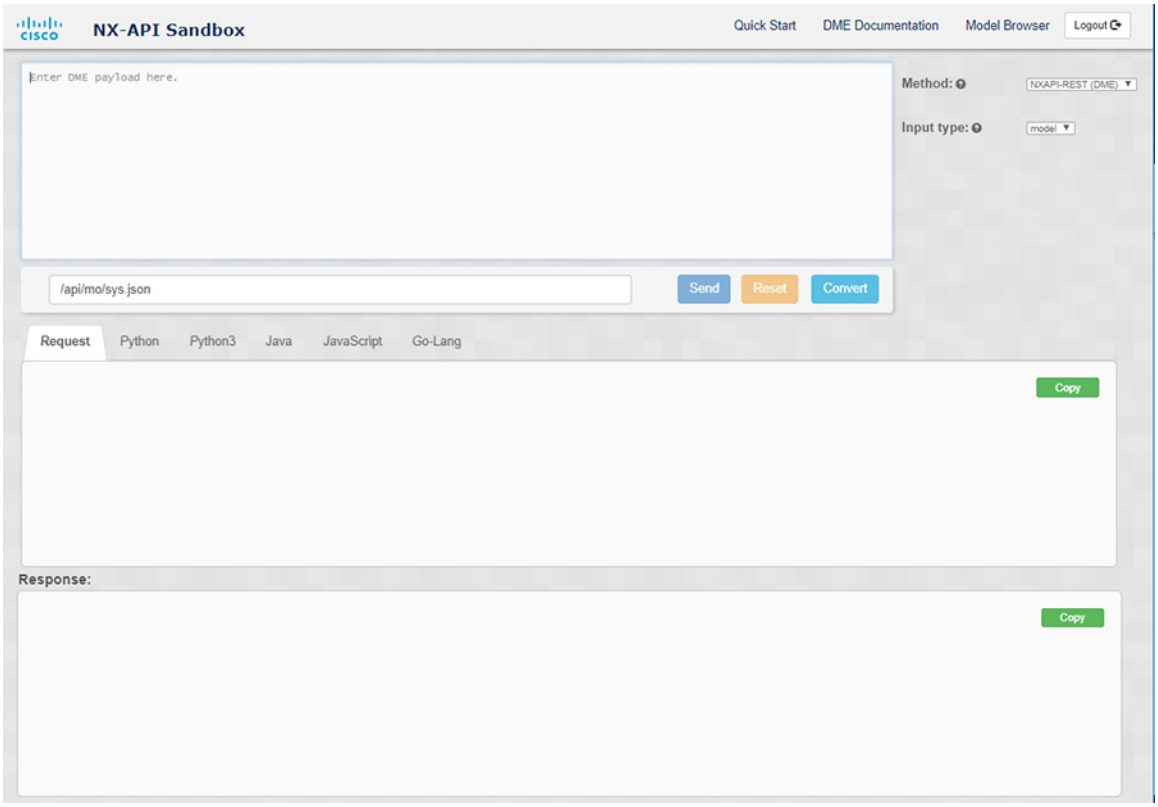
  For additional details, such as response codes and security methods, see the chapter *NX-API CLI*.

- The top-right corner of the Cisco NX-API Developer Sandbox contains links for additional information. The links that appear depend on the **Method** you choose. The links that appear for the NXAPI-REST (DME) method:
  - **NX-API References**—Enables you to access additional NX-API documentation.
  - **DME Documentation**—Enables you to access the NX-API DME Model Reference page.
  - **Model Browser**—Enables you to access Visore, the Model Browser. Note that you might have to manually enter the IP address for your switch to access the Visore page:

    `https://management-ip-address/visore.html`.

**Step 1** Click the **Method** drop-down list and choose **NXAPI-REST (DME)**.

**Example:**




**Step 2** Click the **Input Type** drop-down list and choose **model**.

**Step 3** Enter the designated name (DN) that corresponds to the payload in the field above the Request pane.

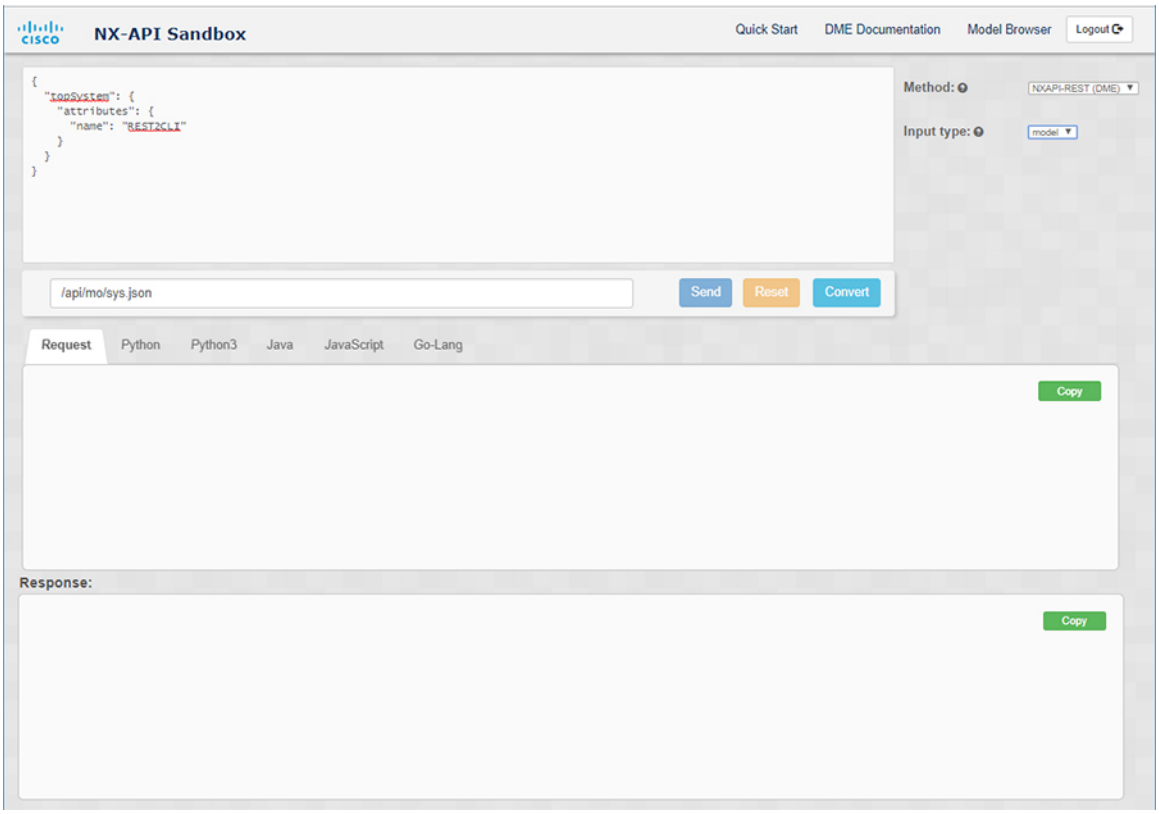
**Step 4** Enter the payload in the Command pane.

**Step 5** Click **Convert**.

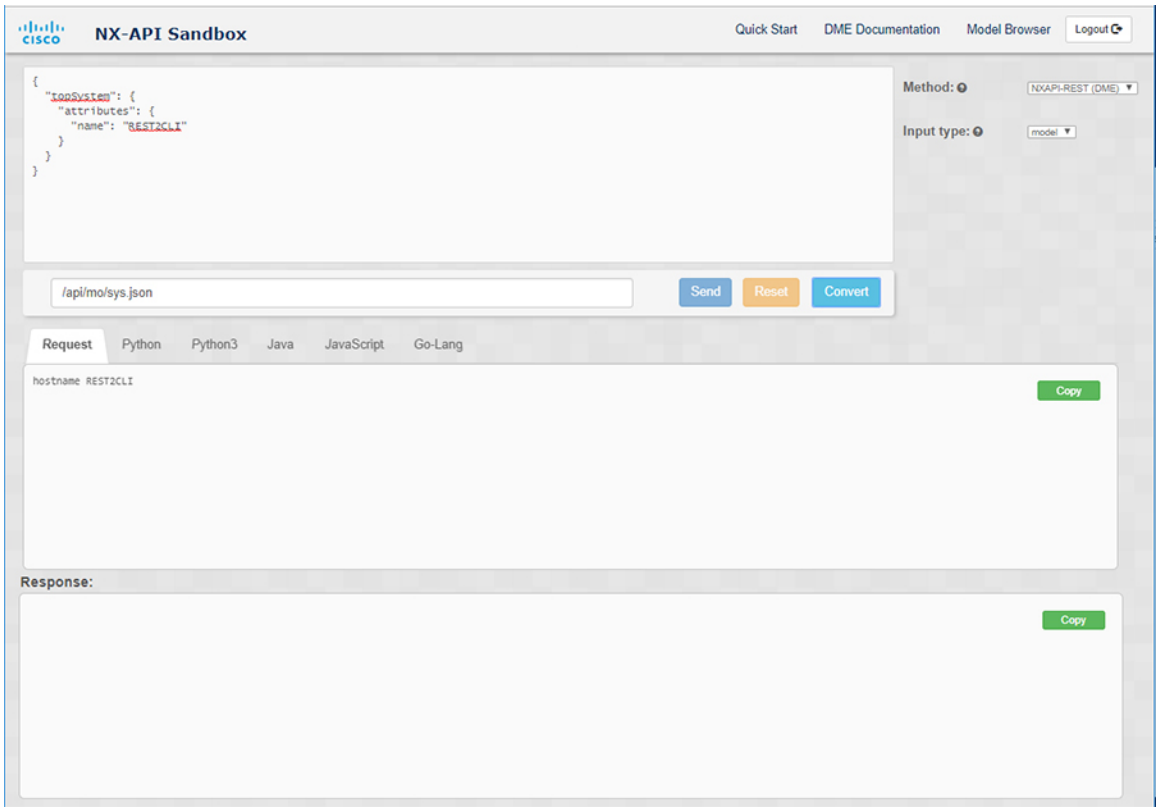
**Example:**

For this example, the DN is **`/api/mo/sys.json`** and the NX-API REST payload is:

```
{
  "topSystem": {
    "attributes": {
      "name": "REST2CLI"
    }
  }
}
```




When you click on the **Convert** button, the CLI equivalent appears in the **CLI** pane as shown in the following image.


NX-API Sandbox
Quick Start
DME Documentation
Model Browser
Logout
{
"topSystem": {
"attributes": {
"name": "REST2CLI"
}
}
}
Method:
NXAPI-REST (DME)
Input type:
model
/api/mo/sys.json
Send
Reset
Convert
Request
Python
Python3
Java
JavaScript
Go-Lang
hostname REST2CLI
Copy
Response:
Copy

**Note** The Cisco NX-API Developer Sandbox cannot convert all payloads into equivalent CLIs, even if the sandbox converted the CLIs to NX-API REST payloads. The following is a list of possible sources of error that can prevent a payload from completely converting to CLI commands:

***Table 6: Sources of REST2CLI Errors***

| Payload Issue | Result |
|---|---|
| The payload contains an attribute that does not exist in the MO.<br>Example:<br>`api/mo/sys.json`<br>`{ "topSystem": { "children": [ { "interfaceEntity": { "children": [ { "l1PhysIf": { "attributes": { "id": "eth1/1", "fakeattribute": "totallyFake" } } } ] } } ] } }` | The **Error** pane will return an error related to the attribute.<br>Example:<br>**CLI**<br>**Error** `unknown attribute 'fakeattribute' in element 'l1PhysIf'` |
| The payload includes MOs that aren't yet supported for conversion:<br>Example:<br>`api/mo/sys.json`<br>`{ "topSystem": { "children": [ { "dhcpEntity": { "children": [ { "dhcpInst": { "attributes": { "SnoopingEnabled": "yes" } } } ] } } ] } }` | The **Error** Pane will return an error related to the unsupported MO.<br>Example:<br>**CLI**<br>**Error** `The entire subtree of "sys/dhcp" is not converted.` |

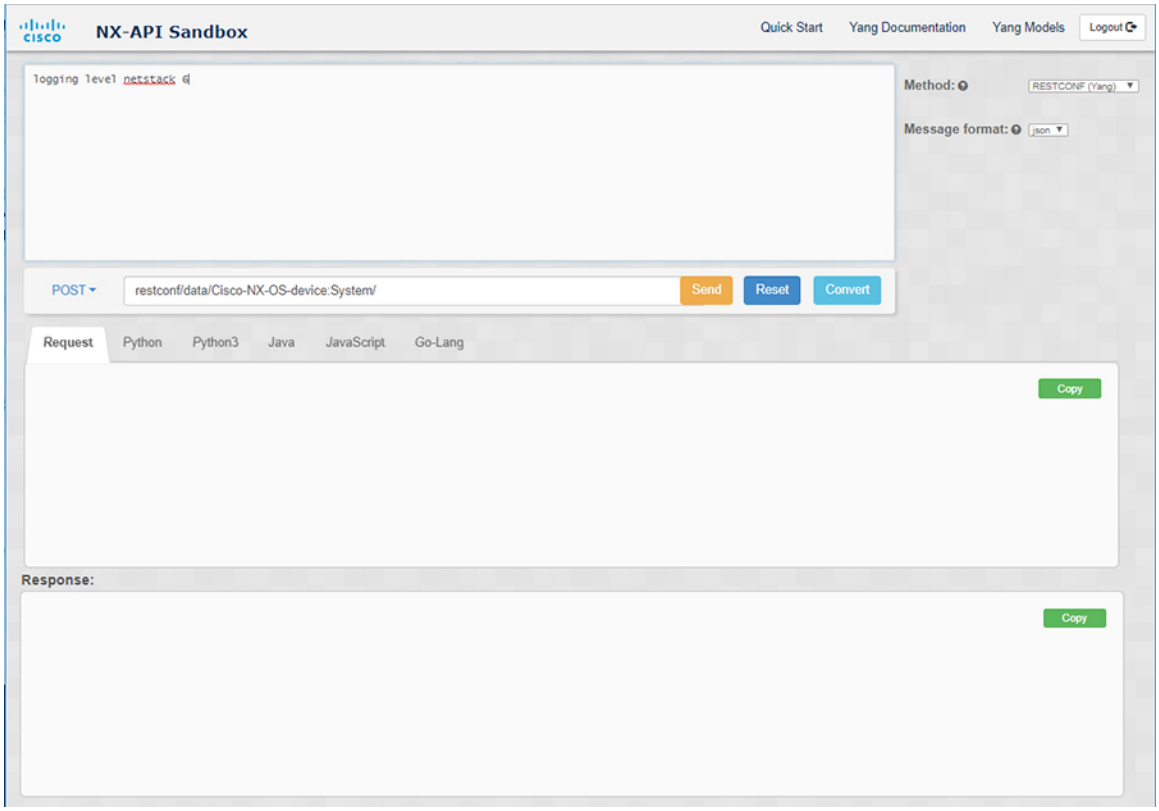
# Using the Developer Sandbox to Convert from RESTCONF to json or XML

**Tip**

- Online help is available by clicking the help icon (**?**) in the upper-right corner of the Cisco NX-API Developer Sandbox window.
- Click on the **Yang Documentation** link in the upper right corner of the Sandbox window to go to the Model Driven Programmability with Yang page.
- Click on the **Yang Models** link in the upper right corner of the Sandbox window to access the YangModels GitHub site.

**Step 1** Click the **Method** drop-down list and choose **RESTCONF (Yang)**.

**Example:**




**Step 2** Click **Message format** and choose either **json** or **xml**.

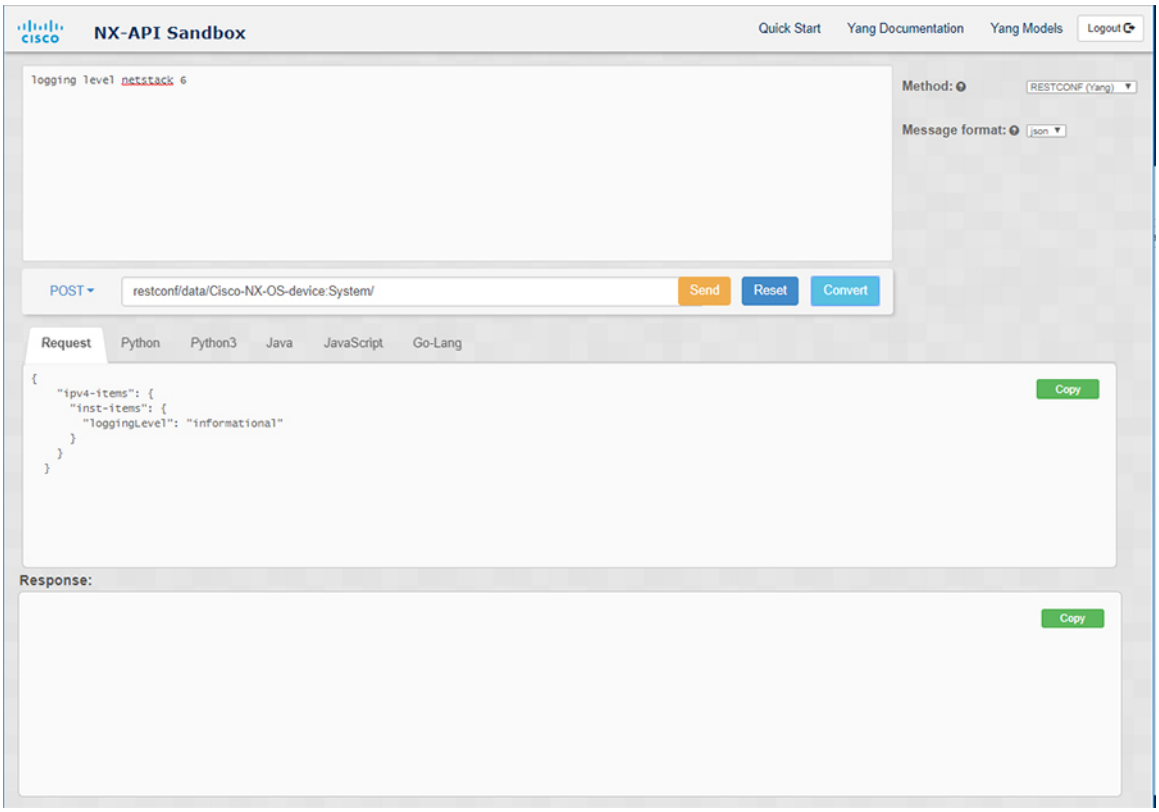
**Step 3** Enter a command in the text entry box in the top pane.

**Step 4** Choose a message format.

**Step 5** Click **Convert**.

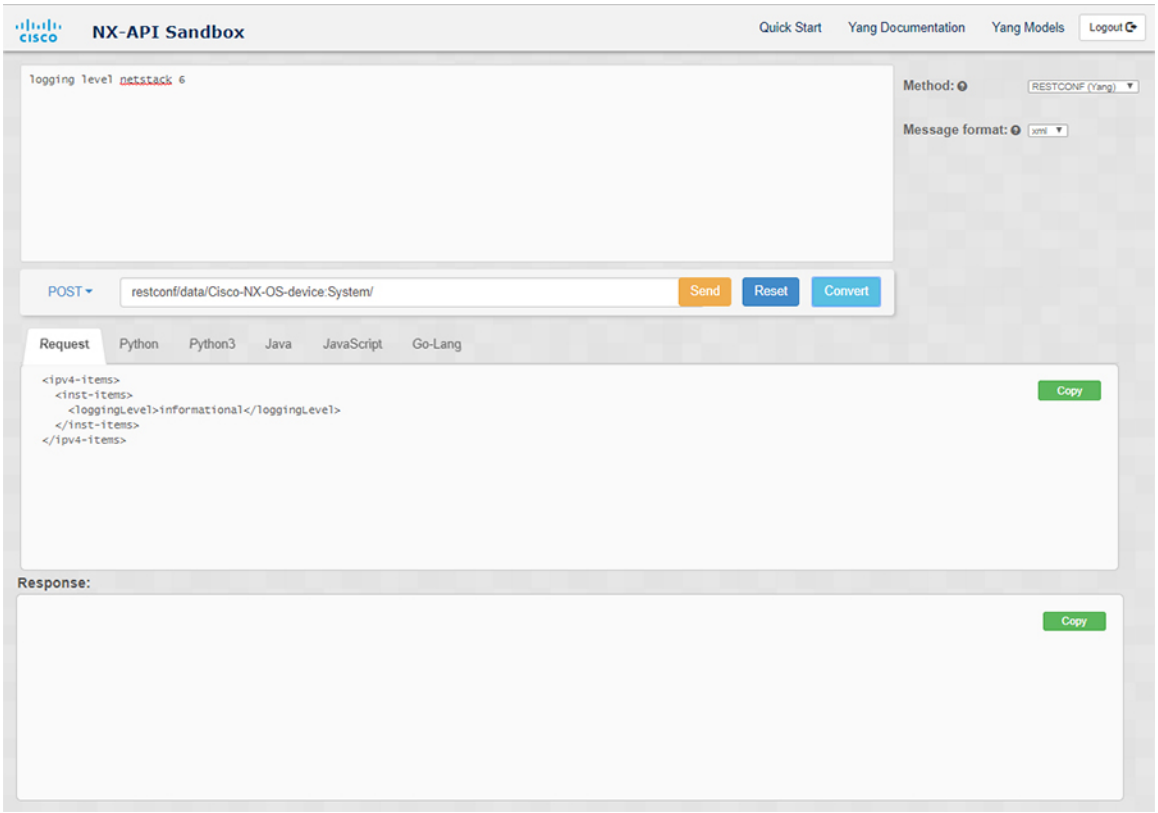
**Example:**

For this example, the command is **`logging level netstack 6`** and the message format is json:




**Example:**

For this example, the command is **`logging level netstack 6`** and the message format is xml:




**Step 6** You can also convert the request into the following formats by clicking on the appropriate tab in the **Request** pane:

- Python
- Python3
- Java
- JavaScript
- Go-Lang

**Note** The Java-generated script does not work if you choose the PATCH option from the drop-down menu in the area above the Request tab. This is a known limitation with Java and is expected behavior.