



NX-API 開発者サンドボックス

- [NX-API 開発者サンドボックス:9.2 \(2\) より前の NX-OS リリース \(1 ページ\)](#)
- [NX-API 開発者サンドボックス : NX-OS リリース 9.2 \(2\) 以降 \(14 ページ\)](#)

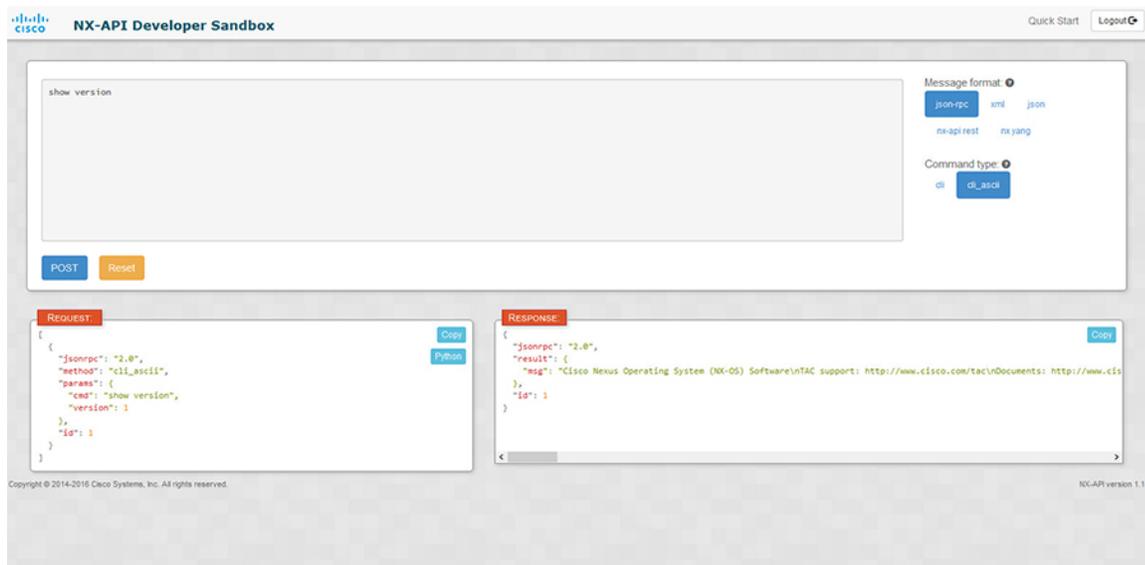
NX-API 開発者サンドボックス:9.2 (2) より前の NX-OS リリース

About the NX-API デベロッパー サンドボックス

NX-API Developer Sandbox は、スイッチでホストされる Web フォームです。NX-OS CLI コマンドを同等の XML または JSON ペイロード。

図に示すように、Web フォームは 3 つのペイン (コマンド (上部ペイン)、要求、および応答) を持つ 1 つの画面です。

図 1: リクエストと出力応答の例を含む NX-API デベロッパー サンドボックス



コマンドペインのコントロールを使用すると、サポートされている API のメッセージフォーマット (NX-API REST など) とコマンドタイプ (XML や JSON など) を選択できます。使用可能なコマンドタイプオプションは、選択したメッセージフォーマットによって異なります。

コマンドペインに 1 つ以上の CLI コマンドを入力するか貼り付けると、Web フォームはコマンドを API ペイロードに変換し、構成エラーをチェックし、結果のペイロードを要求ペインに表示します。次に、コマンドペインの POST ボタンを使用して、ペイロードをサンドボックスからスイッチに直接送信することを選択した場合、応答ペインに API 応答が表示されます。

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.
- When you use XML or JSON as the Message Type for CLI input, you can use semicolon to separate multiple commands on the same line. However, when you use JSON RPC as the Message Type for CLI input, you cannot enter multiple commands on the same line and separate them with a semicolon (;).

For example, assume that you want to send **show hostname** and **show clock** commands through JSON RPC as the following.

In the Sandbox, you enter the CLIs as follows.

```
show hostname ; show clock
```

In the JSON RPC request, the input is formatted as follows.

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

When you send the request, the response returns the following error.

```
{
  "jsonrpc": "2.0",
  "error": {
    "code": -32602,
    "message": "Invalid params",
    "data": {
      "msg": "Request contains invalid special characters"
    }
  },
  "id": 1
}
```

This situation occurs because the Sandbox parses each command in a JSON RPC request as individual items and assigns an ID to each. When using JSON RPC requests, you cannot use internal punctuation to separate multiple commands on the same line. Instead, enter each command on a separate line and the request completes successfully.

Continuing with the same example, enter the commands as follows in the NX-API CLI.

```
show hostname
show clock
```

In the request, the input is formatted as follows.

```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show hostname",
      "version": 1
    },
    "id": 1
  },
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show clock",
      "version": 1
    },
    "id": 2
  }
]
```

```

    }
  ]
  The response completes successfully.
  [
    {
      "jsonrpc": "2.0",
      "result": {
        "body": {
          "hostname": "switch-1"
        }
      },
      "id": 1
    },
    {
      "jsonrpc": "2.0",
      "result": {
        "body": {
          "simple_time": "12:31:02.686 UTC Wed Jul 10 2019\n",
          "time_source": "NTP"
        }
      },
      "id": 2
    }
  ]

```

メッセージフォーマットとコマンドタイプの構成

[メッセージフォーマット (Message Format)] と [コマンドタイプ (Command Type)] は、コマンドペイン (上部ペイン) の右上隅で構成されます。[メッセージフォーマット (Message Format)] で、使用する API プロトコルのフォーマットを選択します。開発者サンドボックスは、次の API プロトコルをサポートしています。

表 1: NX-OS API プロトコル

プロトコル	説明
json-rpc	JSON ペイロードで NX-OS CLI コマンドを配信するために使用できる標準の軽量リモートプロシージャコール (RPC) プロトコル。JSON-RPC 2.0 仕様は、 jsonrpc.org によって概説されています。
xml	XML ペイロードで NX-OS CLI または bash コマンドを配信するための Cisco NX-API 独自のプロトコル。
json	JSON ペイロードで NX-OS CLI または bash コマンドを配信するための Cisco NX-API 独自のプロトコル。
nx-api rest	内部 NX-OS データ管理エンジン (DME) モデルで管理対象オブジェクト (MO) とそのプロパティを操作および読み取るための Cisco NX-API 独自のプロトコル。Cisco Nexus 3000 および 9000 シリーズ NX-API REST SDK の詳細については、 https://developer.cisco.com/site/cisco-nexus-nx-api-references/ を参照してください。

プロトコル	説明
nx yang	構成および状態データ用の YANG (「Yet Another Next Generation」) データモデリング言語。

[メッセージフォーマット (Message Format)] を選択すると、[コマンドタイプ (Command Type)] オプションのセットが [メッセージフォーマット (Message Format)] コントロールのすぐ下に表示されます。[コマンドタイプ (Command Type)] の設定は、入力 CLI を制限でき、[要求 (Request)] と [応答 (Response)] のフォーマットを決定できます。オプションは、選択した [メッセージフォーマット (Message Format)] によって異なります。各 [メッセージフォーマット (Message Format)] について、次の表で [コマンドタイプ (Command Type)] オプションについて説明します。

表 2: コマンドタイプ

メッセージ形式	コマンドタイプ
json-rpc	<ul style="list-style-type: none"> • cli — show または構成コマンド • cli_ascii — show または構成コマンド、フォーマットせずに出力
xml	<ul style="list-style-type: none"> • cli_show — コマンドを表示します。コマンドが XML 出力をサポートしていない場合、エラーメッセージが返されます。 • cli_show_ascii — コマンドを表示、フォーマットせずに出力 • cli_conf — 構成コマンド。対話型の構成コマンドはサポートされていません。 • bash — bash コマンド。ほとんどの非対話型 bash コマンドがサポートされています。 <p>(注) スイッチで bash シェルを有効にする必要があります。</p>

メッセージ形式	コマンドタイプ
json	<ul style="list-style-type: none"> • <code>cli_show</code> — コマンドを表示します。コマンドが XML 出力をサポートしていない場合、エラーメッセージが返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出る • <code>cli_conf</code> — 構成コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — <code>bash</code> コマンド。ほとんどの非対話型 <code>bash</code> コマンドがサポートされています。 <p>(注) スイッチで <code>bash</code> シェルを有効にする必要があります。</p>
nx-api rest	<ul style="list-style-type: none"> • <code>cli</code> — 構成コマンド
nx yang	<ul style="list-style-type: none"> • <code>json</code> — ペイロードに JSON 構造が使用されます • <code>xml</code> — XML 構造がペイロードに使用されます

出力チャンク

大量の `show` コマンド出力を処理するために、一部の NX-API メッセージフォーマットでは、`show` コマンドの出力チャンクがサポートされています。この場合、**[チャンクモードを有効にする (Enable chunk mode)]** チェックボックスが、セッション ID (SID) 入力ボックスとともに **[コマンドタイプ (Command Type)]** コントロールの下に表示されます。

チャンクが有効な場合、応答は複数の「チャンク」で送信され、最初のチャンクが即時のコマンド応答で送信されます。応答メッセージの次のチャンクを取得するには、前の応答メッセージのセッション ID に設定された **SID** を使用して NX-API 要求を送信する必要があります。

デベロッパー サンドボックスを使用

デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する



ヒント オンラインヘルプは、サンドボックス ウィンドウの右上隅にある **[クイックスタート (Quick Start)]** をクリックすると利用できます。

応答コードやセキュリティメソッドなどの詳細については、「NX-API CLI」の章を参照してください。

構成コマンドはサポートされていません。

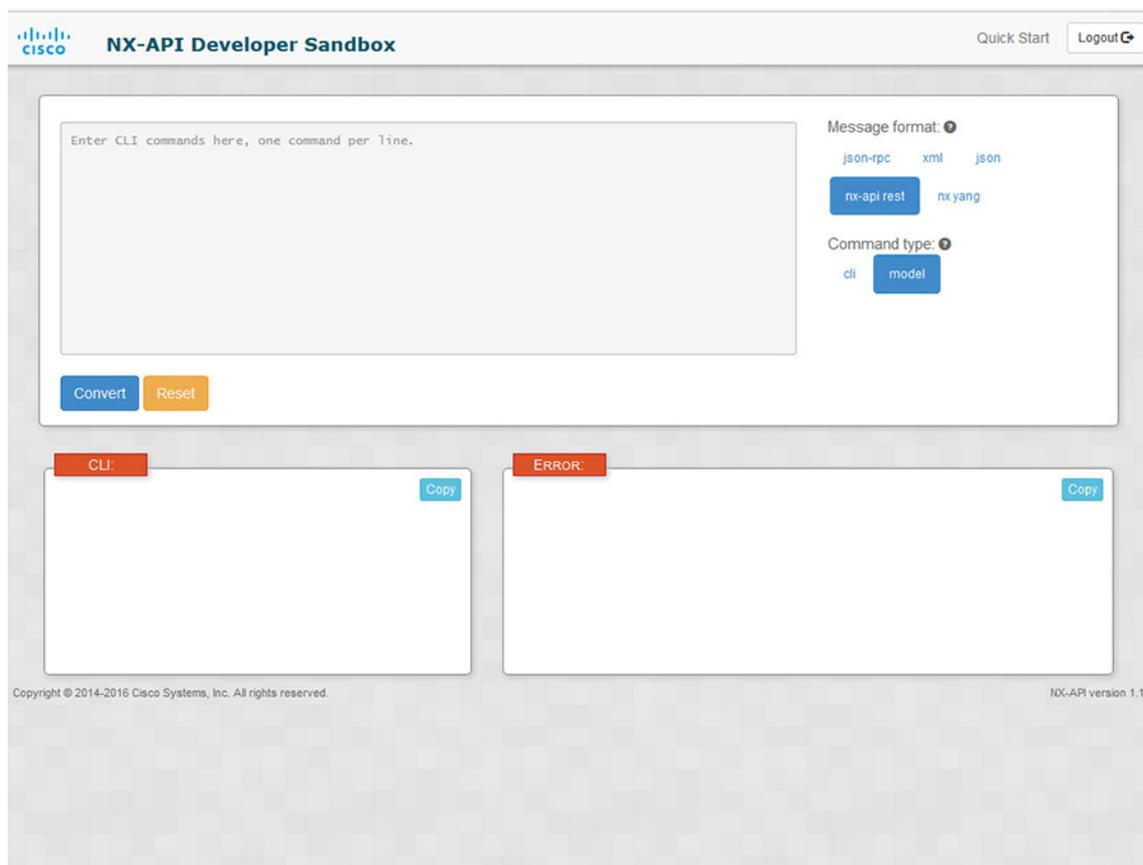
手順

ステップ 1 使用する API プロトコルの[メッセージ形式 (Message Format)]と[コマンドタイプ (Command Type)]を構成します。

詳細な手順については、[メッセージフォーマットとコマンドタイプの構成 \(4 ページ\)](#) を参照してください。

ステップ 2 上部ペインのテキスト エントリ ボックスに、NX-OS CLI 構成コマンドを 1 行に 1 つずつ入力するか貼り付けます。

上部ペインの下部にある[リセット (Reset)]をクリックすると、テキスト エントリ ボックス (および [要求 (Request)] ペインと [応答 (Response)] ペイン)の内容を消去できます。



ステップ 3 トップ ペインの最下部にある [変換 (Convert)] をクリックします。

CLI コマンドに構成エラーが含まれていない場合、ペイロードは [要求 (Request)] ペインに表示されます。エラーが存在する場合は、説明のエラーメッセージが [応答 (Response)] ペインに表示されます。

デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する

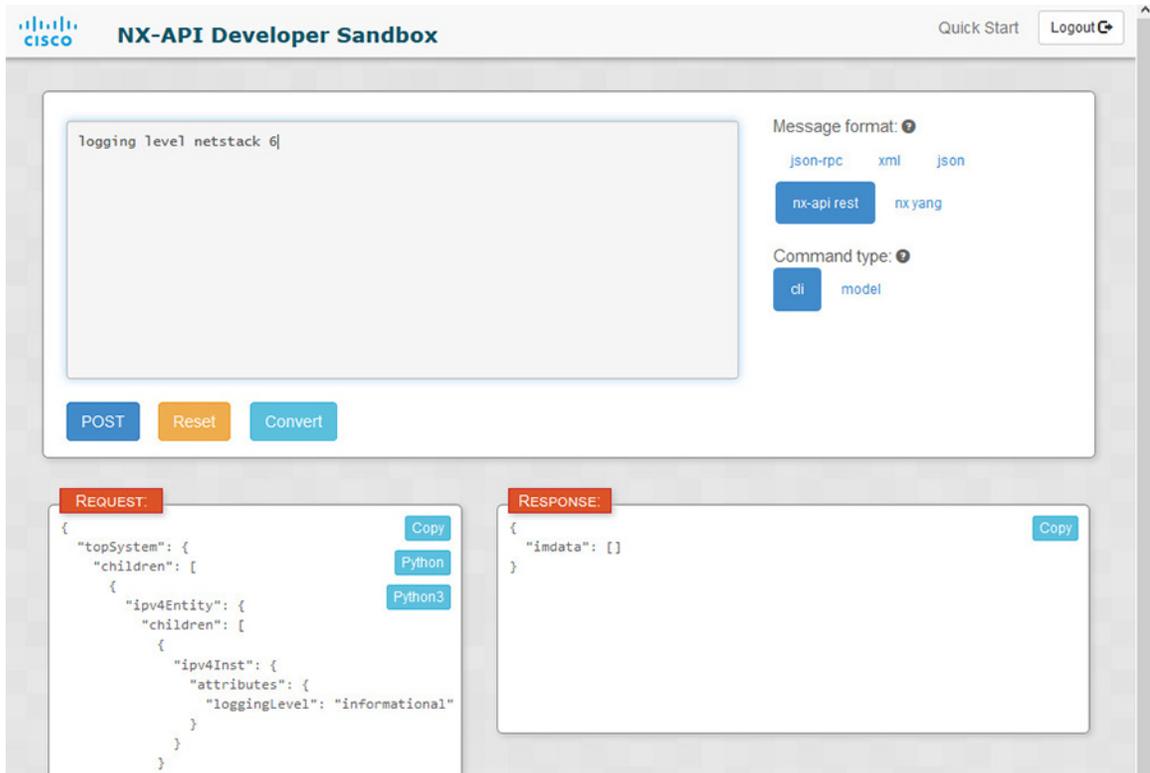
The screenshot displays the NX-API Developer Sandbox interface. At the top, the Cisco logo and 'NX-API Developer Sandbox' title are visible, along with 'Quick Start' and 'Logout' links. The main workspace contains a text editor with a JSON payload for 'api/mo/sys.json'. To the right of the editor are controls for 'Message format' and 'Command type'. Below the editor are 'Convert' and 'Reset' buttons. The interface is divided into two output panels: 'CLI:' and 'ERROR:'. The 'CLI:' panel shows the command 'hostname REST2CLI'. The 'ERROR:' panel is currently empty. At the bottom, a status bar indicates 'Waiting for bam.nr-data.net...' and includes copyright information for Cisco Systems, Inc. (2014-2016) and 'NX-API version 1.1'.

ステップ 4 [リクエスト (Request)] ペインに有効なペイロードが表示されている場合は、**POST** をクリックして、ペイロードを API 呼び出しとしてスイッチに送信できます。

スイッチからのレスポンスは **[Response (応答)]** ペインに表示されます。

警告

POST をクリックすると、コマンドがスイッチにコミットされ、構成または状態が変更される可能性があります。



ステップ5 ペインで[コピー (Copy)]をクリックすると、[要求 (Request)]ペインまたは[応答 (Response)]ペインの格納ファイルをクリップボードにコピーできます。

ステップ6 [リクエスト (Request)]ペインでPythonをクリックすると、クリップボード上のリクエストのPython 導入を取得できます。

デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換



ヒント オンラインヘルプは、サンドボックスウィンドウの右上隅にある[クイックスタート (Quick Start)]をクリックすると利用できます。

応答コードやセキュリティメソッドなどの詳細については、「NX-API CLI」の章を参照してください。

手順の概要

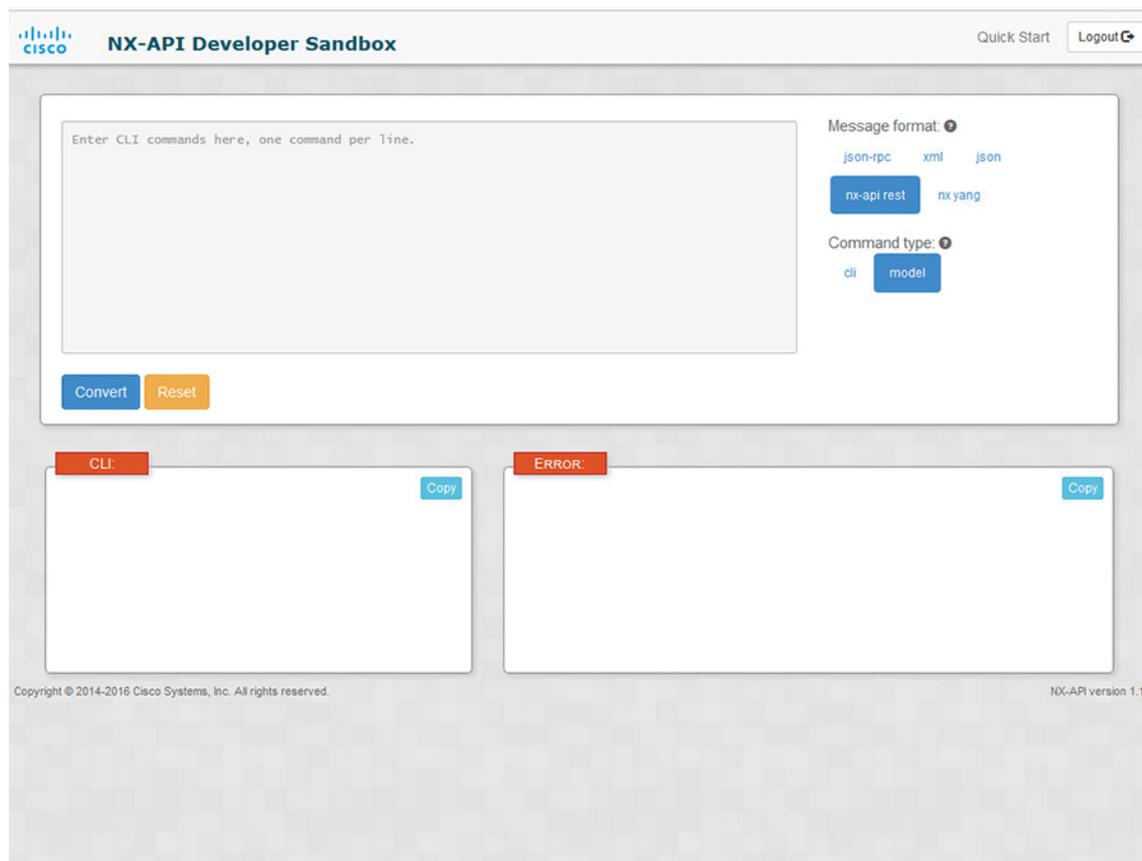
1. メッセージフォーマットとして `nx-api rest` を選択し、コマンドタイプとして `model` を選択します。
2. 上部ペインのテキスト入力ボックスに DN とペイロードを入力します。次に、上部ペインの下にある [変換 (Convert)] ボタンをクリックします。

手順の詳細

手順

ステップ1 メッセージフォーマットとして **nx-api rest** を選択し、コマンドタイプとして **model** を選択します。

例：



ステップ2 上部ペインのテキスト入力ボックスにDNとペイロードを入力します。次に、上部ペインの下にある[変換 (Convert)] ボタンをクリックします。

例：

この例では、DNは **/api/mo/sys.json** であり、NX-API REST ペイロードは次のとおりです：

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

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 contains a code editor with the following JSON payload:

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

Below the code editor are "Convert" and "Reset" buttons. To the right, there are two dropdown menus: "Message format:" with options "json-rpc", "xml", "json", and "nx-api rest" (selected); and "Command type:" with options "cli" and "model" (selected). Below these are two output panes: "CLI:" and "ERROR:". Both panes are currently empty, with "Copy" buttons in their top right corners. At the bottom left, there is a status bar that says "Waiting for bam.nr-data.net...". At the bottom right, it says "NX-API version 1.1".

[変換 (Convert)] ボタンをクリックすると、次の図に示すように、同等の CLI が CLI ペインに表示されます。

The screenshot shows the NX-API Developer Sandbox interface. At the top, there is a Cisco logo and the title "NX-API Developer Sandbox". On the right, there are links for "Quick Start" and "Logout". The main area is divided into two sections. The top section contains a text area with a REST payload:

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

Below the text area are "Convert" and "Reset" buttons. To the right, there are settings for "Message format:" (with options: json-rpc, xml, json, nx-api rest, nx.yang) and "Command type:" (with options: cli, model). The bottom section is split into two panels. The left panel, labeled "CLI:", shows the converted command: "hostname REST2CLI" with a "Copy" button. The right panel, labeled "ERROR:", is empty with 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".

(注)

デベロッパーサンドボックスは、サンドボックスが CLI を NX-API REST ペイロードに変換した場合でも、すべてのペイロードを同等の CLI に変換することはできません。以下は、ペイロードが CLI コマンドに完全に変換するのを妨げる可能性のあるエラーの原因のリストです。

表 3: REST2CLI エラーの原因

ペイロードの問題	結果
<p>ペイロードに、MO に存在しない属性が含まれています。</p> <p>例：</p> <pre>api/mo/sys.json { "topSystem": { "children": [{ "interfaceEntity": { "children": [{ "l1PhysIf": { "attributes": { "id": "eth1/1", "fakeattribute": "totallyFake" } } }] } }] } }</pre>	<p>[エラー (Error)] ペインは、属性に関連するエラーを返します。</p> <p>例：</p> <p>CLI</p> <p>要素「l1PhysIf」の不明な属性「fakeattribute」の[エラー (Error)]</p>
<p>ペイロードには、変換がまだサポートされていない MO が含まれています。</p> <p>例：</p> <pre>api/mo/sys.json { "topSystem": { "children": [{ "dhcpEntity": { "children": [{ "dhcpInst": { "attributes": { "SnoopingEnabled": "yes" } } }] } }] } }</pre>	<p>[エラー (Error)] ペインは、サポートされていない MO に関連するエラーを返します。</p> <p>例：</p> <p>CLI</p> <p>[エラー (Error)] [「sys/dhcp」のサブツリー全体が変換されていません。(The entire subtree of "sys/dhcp" is not converted.)]</p>

NX-API 開発者サンドボックス : NX-OS リリース 9.2 (2) 以降

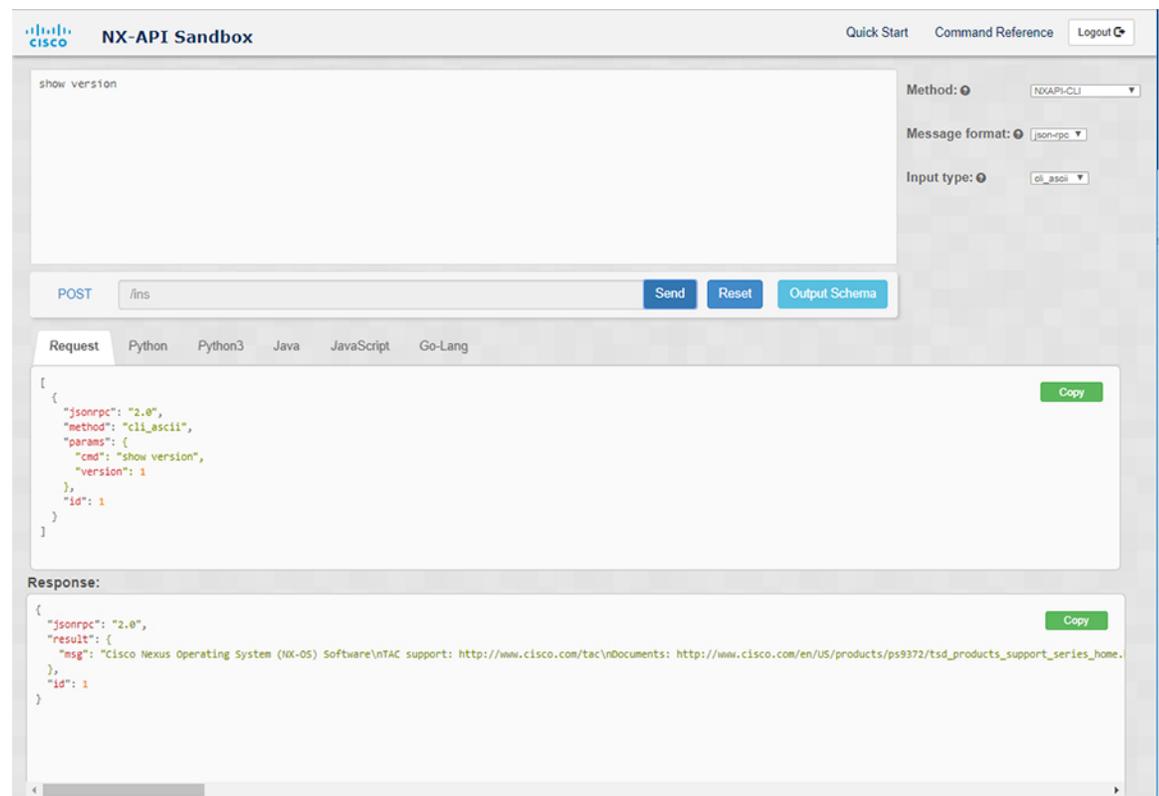
About the NX-API デベロッパー サンドボックス

Cisco NX-API Developer Sandbox は、スイッチでホストされる Web フォームです。NX-OS CLI コマンドを同等の XML または JSON ペイロードに変換し、NX-API REST ペイロードを同等の CLI に変換します。

Web フォームは、次の図に示すように、コマンド（上部のペイン）、要求（中央のペイン）、および応答（下部のペイン）の 3 つのペインを持つ 1 つの画面です。指定名（DN）フィールドは、コマンド ペインとリクエスト ペインの間にあります（下図の **POST** と送信オプションの間にあります）。

リクエスト ペインにも一連のタブがあります。各タブは、**Python**、**Python3**、**Java**、**JavaScript**、**Go-Lang** の異なる言語を表します。各タブでは、それぞれの言語でリクエストを表示できます。たとえば、CLI コマンドを XML または JSON ペイロードに変換した後、**[Python]** タブをクリックして、スクリプトの作成に使用できる Python でのリクエストを表示します。

図 2: リクエストと出力応答の例を含む NX-API デベロッパー サンドボックス



コマンドペインのコントロールを使用すると、NX-API REST などのサポートされている API、モデル（ペイロード）や CLI などの入力タイプ、および XML や JSON などのメッセージ形式を選択できます。使用可能なオプションは、選択した方法によって異なります。

NXAPI-REST（DME）メソッドを選択し、1つ以上の CLI コマンドをコマンドペインに入力するか貼り付けて、**[変換]** をクリックすると、Web フォームはコマンドを REST API ペイロードに変換し、構成エラーをチェックし、要求ペインに結果のペイロードを表示します。次に、ペイロードをサンドボックスからスイッチに直接送信することを選択した場合（**POST** オプションを選択して **[SEND]** をクリック）、**[応答]** ペインに API 応答が表示されます。詳細については、[デベロッパーサンドボックスを使用して CLI コマンドを REST ペイロードに変換する（21 ページ）](#) を参照してください。

逆に、Cisco NX-API Developer Sandbox はペイロードの設定エラーをチェックし、対応する CLI を **[応答]** ペインに表示します。詳細については、「[デベロッパーサンドボックスを使用した REST ペイロードから CLI コマンドへの変換（24 ページ）](#)」を参照してください。

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.
- When you use XML or JSON as the Message Type for CLI input, you can use semicolon to separate multiple commands on the same line. However, when you use JSON RPC as the Message Type for CLI input, you cannot enter multiple commands on the same line and separate them with a semicolon (;).

For example, assume that you want to send **show hostname** and **show clock** commands through JSON RPC as the following.

In the Sandbox, you enter the CLIs as follows.

```
show hostname ; show clock
```

In the JSON RPC request, the input is formatted as follows.

```
[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show hostname ; show clock",
      "version": 1
    }
  }
]
```

```

    },
    "id": 1
  }
]

```

When you send the request, the response returns the following error.

```

{
  "jsonrpc": "2.0",
  "error": {
    "code": -32602,
    "message": "Invalid params",
    "data": {
      "msg": "Request contains invalid special characters"
    }
  },
  "id": 1
}

```

This situation occurs because the Sandbox parses each command in a JSON RPC request as individual items and assigns an ID to each. When using JSON RPC requests, you cannot use internal punctuation to separate multiple commands on the same line. Instead, enter each command on a separate line and the request completes successfully.

Continuing with the same example, enter the commands as follows in the NX-API CLI.

```

show hostname
show clock

```

In the request, the input is formatted as follows.

```

[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show hostname",
      "version": 1
    },
    "id": 1
  },
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show clock",
      "version": 1
    },
    "id": 2
  }
]

```

The response completes successfully.

```

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

```

```

    "result": {
      "body": {
        "simple_time": "12:31:02.686 UTC Wed Jul 10 2019\n",
        "time_source": "NTP"
      }
    },
    "id": 2
  }
]

```

メッセージフォーマットと入カタイプの構成

メソッド、メッセージ形式、および入カタイプは、コマンドペイン（上部のペイン）の右上隅で構成されます。[メソッド]で、使用するAPIプロトコルの形式を選択します。Cisco NX-API Developer Sandbox は、次のAPIプロトコルをサポートしています。

表 4: NX-OS API プロトコル

プロトコル	説明
NXAPI-CLI	XML または JSON ペイロードで NX-OS CLI または bash コマンドを配信するための Cisco NX-API 独自のプロトコル。
NXAPI-REST (DME)	<p>内部 NX-OS データ管理エンジン（DME）モデルで管理対象オブジェクト（MO）とそのプロパティを操作および読み取るための Cisco NX-API 独自のプロトコル。NXAPI-REST（DME）プロトコルは、次の方法から選択できるドロップダウンリストを表示します。</p> <ul style="list-style-type: none"> • POST • GET • PUT • DELETE <p>Cisco Nexus 3000 および 9000 シリーズ NX-API REST SDK の詳細については、https://developer.cisco.com/site/cisco-nexus-nx-api-references/ を参照してください。</p>

プロトコル	説明
RESTCONF (Yang)	<p>構成および状態データ用の YANG (「Yet Another Next Generation」) データモデリング言語。</p> <p>RESTCONF (Yang) プロトコルは、次の方法から選択できるドロップダウンリストを表示します。</p> <ul style="list-style-type: none"> • POST • GET • PUT • PATCH • DELETE

メソッドを選択すると、メッセージ形式または入力タイプのオプションのセットがドロップダウンリストに表示されます。メッセージ形式は、入力 CLI を制約し、要求と応答の形式を決定できます。オプションは、選択したメソッドによって異なります。

次の表では、各メッセージ形式の入力/コマンドタイプオプションについて説明します。

表 5: コマンドタイプ

方法	メッセージ形式	入力/コマンドタイプ
NXAPI-CLI	json-rpc	<ul style="list-style-type: none"> • cli — show または構成コマンド • cli_ascii — show または構成コマンド、フォーマットせずに出力 • cli_array — show コマンド。cli に似ていますが、cli_array を使用すると、データは 1 つの要素のリスト、または角括弧 [] で囲まれたアレイとして返されます。

方法	メッセージ形式	入力/コマンドタイプ
NXAPI-CLI	xml	<ul style="list-style-type: none"> • <code>cli_show</code> — コマンドを表示します。コマンドがXML出力をサポートしていない場合、エラーメッセージが返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出力 • <code>cli_conf</code> — 構成 コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — <code>bash</code> コマンド。ほとんどの非対話型 <code>bash</code> コマンドがサポートされています。 <p>(注) スイッチで <code>bash</code> シェルを有効にする必要があります。</p>
NXAPI-CLI	json	<ul style="list-style-type: none"> • <code>cli_show</code> — コマンドを表示します。コマンドがXML出力をサポートしていない場合、エラーメッセージが返されます。 <p>(注) Cisco NX-OS リリース 9.3(3) 以降では、<code>cli_show</code> コマンドよりも <code>cli_show_array</code> コマンドが推奨されます。</p> <ul style="list-style-type: none"> • <code>cli_show_array</code> — <code>show</code> コマンド <code>cli_show</code> に似ていますが、<code>cli_show_array</code> を使用すると、データは角括弧 [] で囲まれた1つの要素のリストまたは配列として返されます。 • <code>cli_show_ascii</code> — コマンドを表示、フォーマットせずに出力 • <code>cli_conf</code> — 構成 コマンド。対話型の構成コマンドはサポートされていません。 • <code>bash</code> — <code>bash</code> コマンド。ほとんどの非対話型 <code>bash</code> コマンドがサポートされています。 <p>(注) スイッチで <code>bash</code> シェルを有効にする必要があります。</p>

方法	メッセージ形式	入力/コマンドタイプ
NXAPI-REST (DME)		<ul style="list-style-type: none"> • cli — CLI からモデルへの変換 • model — モデルから CLI への変換。
RESTCONF (Yang)	<ul style="list-style-type: none"> • json — ペイロードに JSON 構造が使用されます • xml — XML 構造がペイロードに使用されます 	

出力チャンク

JSON および XML NX-API メッセージ形式を使用すると、10 MB のチャンクで大きな show コマンド応答を受信できます。受信すると、チャンクが連結されて、有効な JSON オブジェクトまたは XML 構造が作成されます。出力チャンクを示すサンプルスクリプトを表示するには、次のリンクをクリックし、リリース 9.3x に対応するディレクトリを選択します：[Cisco NX-OS NXAPI](#)。



- (注) チャンク JSON モードの場合、ブラウザまたは Python スクリプト パーツは有効な JSON 出力を提供しません（終了タグはありません）。チャンク モードを使用して有効な JSON を取得するには、ディレクトリで提供されるスクリプトを使用します。

即時のコマンド応答で最初のチャンクを受け取ります。これには、セッション ID を含む **sid** フィールドも含まれます。次のチャンクを取得するには、前のチャンクのセッション ID を **[SID]** テキストボックスに入力します。**sid** フィールドの **eoc**（コンテンツの終わり）値で示される最後の応答に到達するまで、プロセスを繰り返します。

チャンク モードは、JSON または XML フォーマット タイプおよび **cli_show**、**cli_show_array**、または **cli_show_ascii** コマンドタイプで **NXAPI-CLI** メソッドを使用する場合に使用できます。チャンク モードの設定の詳細については、チャンク モード フィールドの表を参照してください。



- (注) NX-API は、最大 2 つのチャンク セッションをサポートします。

表 6: チャンク モード フィールド

フィールド名	説明
チャンク モードを有効にする	[チャンク モードを有効にする (Enable Chunk Mode)] チェックボックスをクリックしてチェックマークを付けると、チャンクが有効になります。チャンク モードを有効にすると、10 MB を超える応答は、最大 10 MB のサイズの複数のチャンクで送信されます。
SID	<p>応答メッセージの次のチャンクを取得するには、SID テキストボックスに前の応答のセッション ID を入力します。</p> <p>(注) 使用できる文字は英数字と「_」のみです。無効な文字はエラーを受け取ります。</p>

デベロッパー サンドボックスを使用

デベロッパー サンドボックスを使用して CLI コマンドを REST ペイロードに変換する



ヒント

- Cisco NX-API デベロッパー サンドボックス ウィンドウの右上隅にあるフィールド名の横にあるヘルプアイコン (?) をクリックすると、オンラインヘルプを利用できます。
- 応答コードやセキュリティ メソッドなどの詳細については、*NX-API CLI* の章を参照してください。
- 構成コマンドはサポートされていません。

Cisco NX-API Developer Sandbox を使用すると、CLI コマンドを REST ペイロードに変換できません。

手順

ステップ 1 [方法 (Method)] ドロップダウン リストをクリックし、**NXAPI-REST (DME)** を選択します。

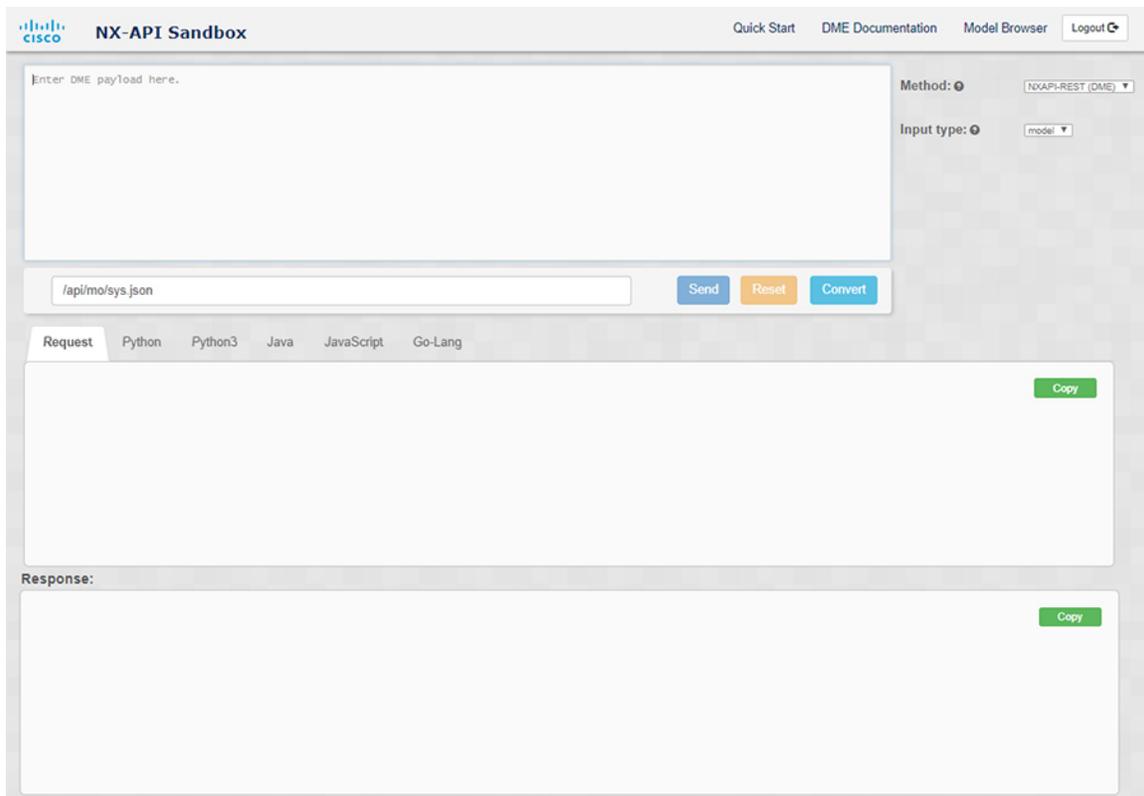
[入カタイプ] ドロップダウン リストが表示されます。

ステップ 2 [入力 (Input)] タイプドロップダウン リストをクリックし、**cli** を選択します。

ステップ 3 上部ペインのテキスト エントリ ボックスに、NX-OS CLI 構成コマンドを 1 行に 1 つずつ入力するか貼り付けます。

上部ペインの下部にある [リセット (Reset)] をクリックすると、テキスト エントリ ボックス (および [要求 (Request)] ペインと [応答 (Response)] ペイン) の内容を消去できます。

デベロッパーサンドボックスを使用して CLI コマンドを REST ペイロードに変換する



ステップ 4 [変換 (Convert)] をクリックします。

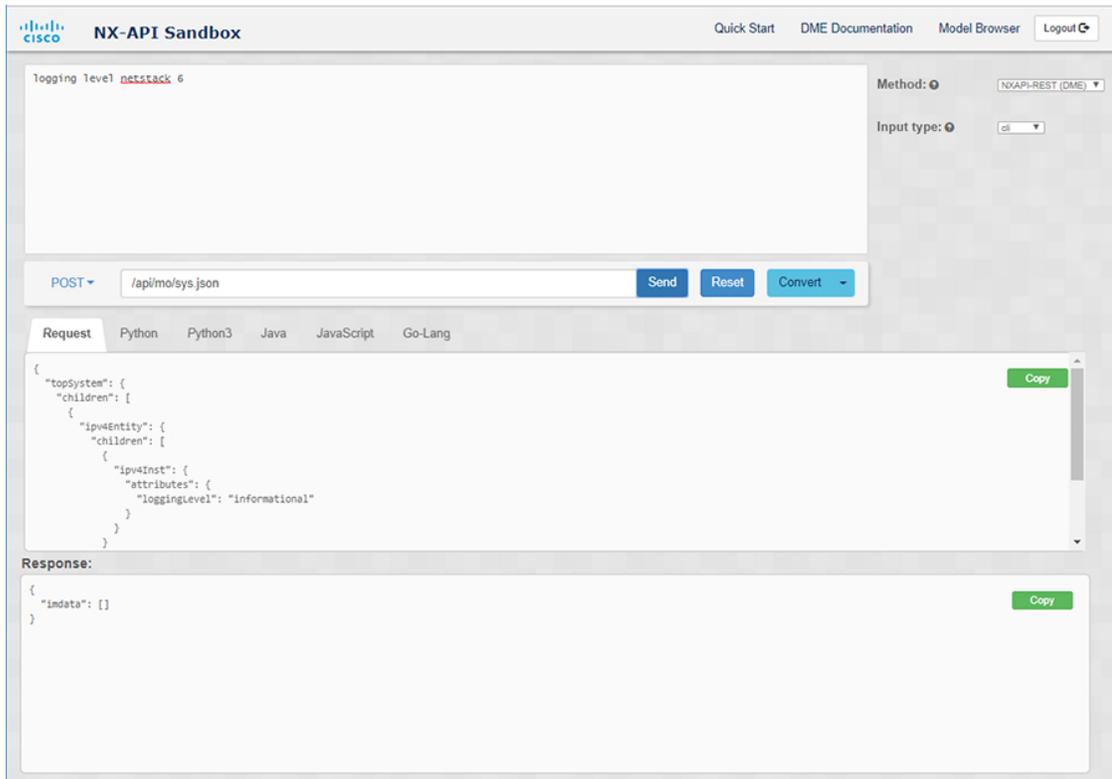
CLI コマンドに構成エラーが含まれていない場合、ペイロードは [要求 (Request)] ペインに表示されます。エラーが存在する場合は、説明のエラーメッセージが [応答 (Response)] ペインに表示されます。

ステップ 5 (オプション) 有効なペイロードを API 呼び出しとしてスイッチに送信するには、[送信 (Send)] をクリックします。

スイッチからのレスポンスは [Response (応答)] ペインに表示されます。

警告

[送信 (Send)] をクリックすると、コマンドがスイッチにコミットされ、構成または状態が変更される可能性があります。



ステップ 6 (オプション) ペイロード内の MO の DN を取得するには：

1. [リクエスト (Request)] ペインから、**POST** を選択します。
2. [変換 (Convert)] ドロップダウンリストをクリックし、[変換 (DN を使用) (Convert (with DN))] を選択します。

ペイロードは、ペイロード内の各 MO に対応する DN を含む **dn** フィールドとともに表示されます。

ステップ 7 (オプション) 新しい構成で現在の構成を上書きする場合：

1. [変換 (Convert)] ドロップダウンリストをクリックし、[変換 (置換用) (Convert (for Replace))] を選択します。[リクエスト (Request)] ペインには、[ステータス (status)] フィールドが[置換 (replace)] ように設定されたペイロードが表示されます。
2. [リクエスト (Request)] ペインから、**POST** を選択します。
3. [送信 (Send)] をクリックします。

現在の構成は、投稿された構成に置き換えられます。たとえば、次の構成で開始するとします：

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

次に、[変換 (置換用) (Convert (for Replace))] を使用して、次の構成を POST します。

デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換

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

mtu 構成が削除され、新しい説明 (testForcr) のみがインターフェイスの下に表示されます。この変更は、**show running-config** と入力すると確認されます。

ステップ 8 (オプション) [リクエスト (Request)] ペインや [応答 (Response)] ペインなどのペインの内容をコピーするには、[コピー (Copy)] をクリックします。それぞれのペインの内容がクリップボードにコピーされます。

ステップ 9 (オプション) リクエストを以下のいずれかのフォーマットに変換するには、[リクエスト (Request)] ペインの適切なタブをクリックします。

- Python
- python3
- Java
- JavaScript
- Go-Lang

デベロッパー サンドボックスを使用した REST ペイロードから CLI コマンドへの変換

Cisco NX-API Developer Sandbox を使用すると、REST ペイロードを対応する CLI コマンドに変換できます。このオプションは、NXAPI-REST (DME) メソッドでのみ使用できます。



ヒント

- Cisco NX-API Developer Sandbox のフィールド名の横にあるヘルプアイコン (?) をクリックすると、オンライン ヘルプを利用できます。ヘルプアイコンをクリックして、それぞれのフィールドに関する情報を取得します。

応答コードやセキュリティ メソッドなどの詳細については、*NX-API CLI* の章を参照してください。

- Cisco NX-API Developer Sandbox の右上隅には、追加情報へのリンクが含まれています。表示されるリンクは、選択した[方法 (Method)]によって異なります。NXAPI-REST (DME) メソッドに表示されるリンク：

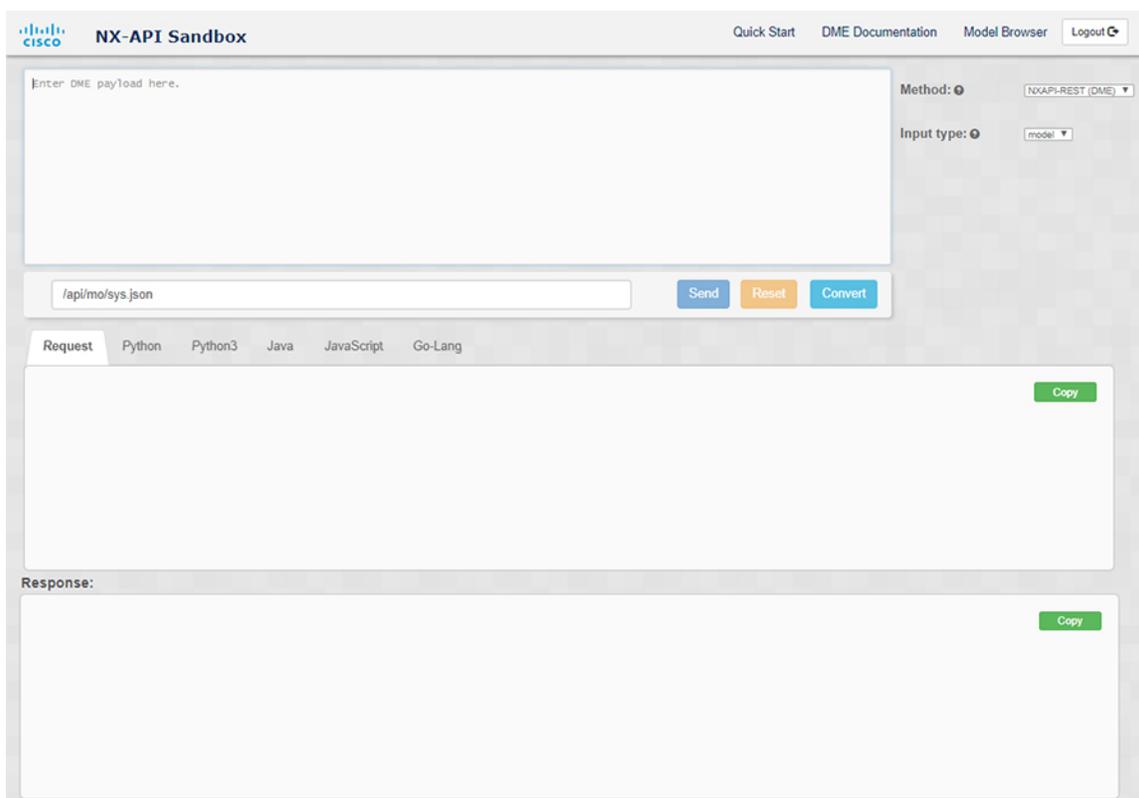
- [NX-API リファレンス (NX-API References)] — 追加の NX-API ドキュメントにアクセスできます。
- [DME ドキュメント (DME Documentation)] — NX-API DME モデル リファレンス ページにアクセスできます。
- [モデル ブラウザ (Model Browser)] — モデル ブラウザである Visore にアクセスできます。Visore ページにアクセスするには、スイッチの IP アドレスを手動で入力する必要がある場合があることに注意してください。

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

手順

ステップ 1 [方法 (Method)] ドロップダウン リストをクリックし、NXAPI-REST (DME) を選択します。

例：



ステップ 2 [タイプを入力 (Input Type)] タイプドロップダウンリストをクリックし、[モデル (model)] を選択します。

ステップ 3 要求ペインの上にあるフィールドに、ペイロードに対応する指定名 (DN) を入力します。

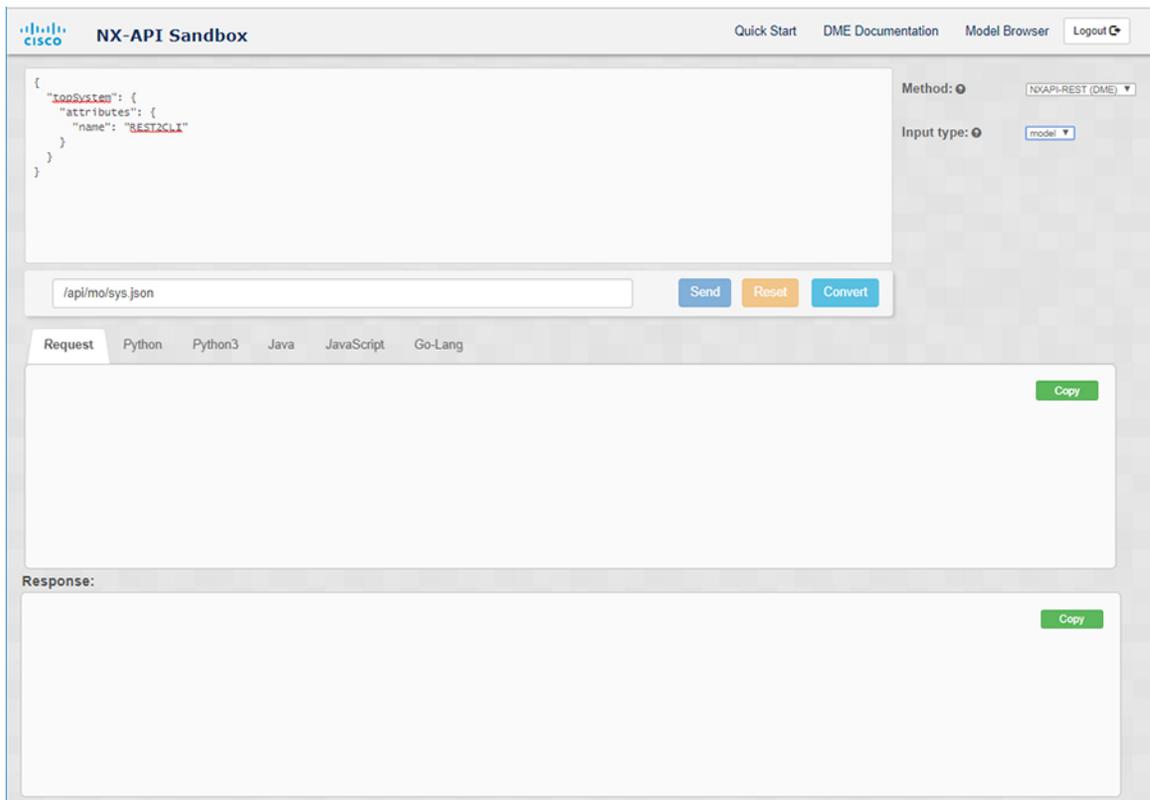
ステップ 4 コマンドペインにペイロードを入力します。

ステップ 5 [変換 (Convert)] をクリックします。

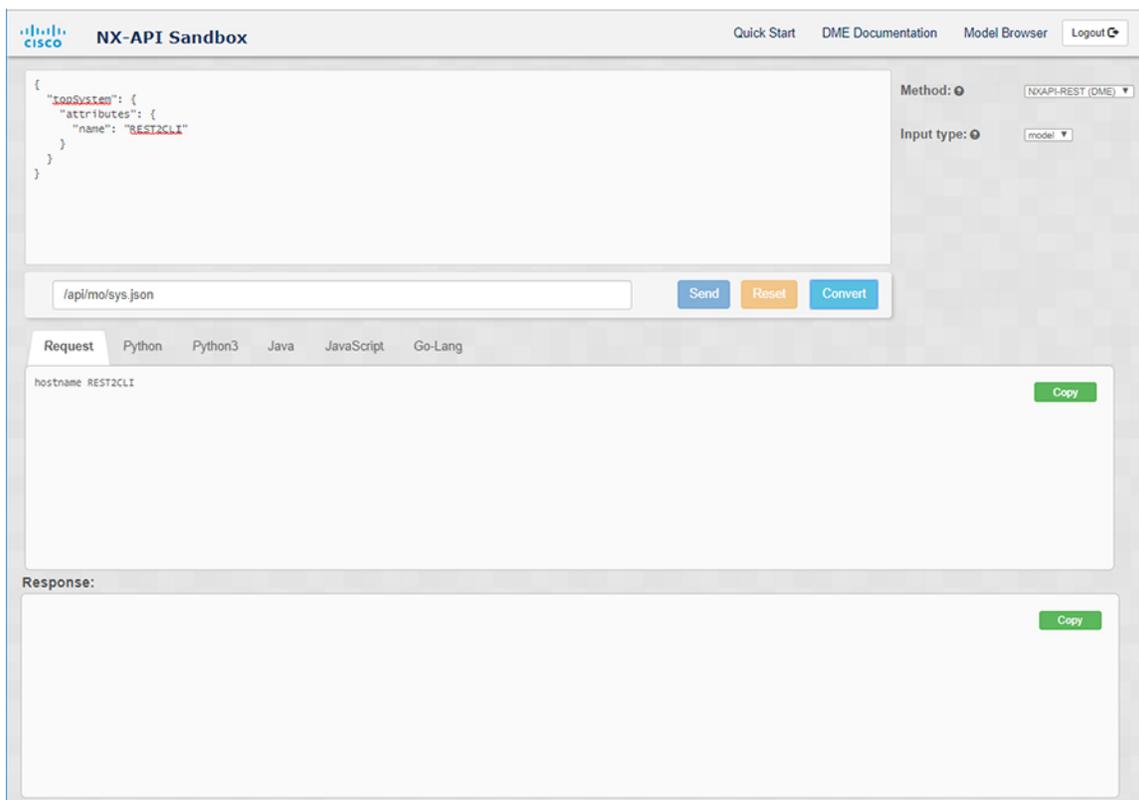
例：

この例では、DN は `/api/mo/sys.json` であり、NX-API REST ペイロードは次のとおりです。

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



[変換 (Convert)] ボタンをクリックすると、次の図に示すように、同等の CLI が CLI ペインに表示されます。



(注)

Cisco NX-API Developer Sandbox は、サンドボックスが CLI を NX-API REST ペイロードに変換した場合でも、すべてのペイロードを同等の CLI に変換できません。以下は、ペイロードが CLI コマンドに完全に変換するのを妨げる可能性のあるエラーの原因のリストです。

表 7: REST2CLI エラーの原因

ペイロードの問題	結果
<p>ペイロードに、MO に存在しない属性が含まれています。</p> <p>例：</p> <pre>api/mo/sys.json { "topSystem": { "children": [{ "interfaceEntity": { "children": [{ "l1PhysIf": { "attributes": { "id": "eth1/1", "fakeattribute": "totallyFake" } } }] } }] } }</pre>	<p>[エラー (Error)] ペインは、属性に関連するエラーを返します。</p> <p>例：</p> <p>CLI</p> <p>要素「l1PhysIf」の不明な属性「fakeattribute」の[エラー (Error)]</p>
<p>ペイロードには、変換がまだサポートされていない MO が含まれています。</p> <p>例：</p> <pre>api/mo/sys.json { "topSystem": { "children": [{ "dhcpEntity": { "children": [{ "dhcpInst": { "attributes": { "SnoopingEnabled": "yes" } } }] } }] } }</pre>	<p>[エラー (Error)] ペインは、サポートされていない MO に関連するエラーを返します。</p> <p>例：</p> <p>CLI</p> <p>[エラー (Error)] [「sys/dhcp」のサブツリー全体が変換されていません。(The entire subtree of "sys/dhcp" is not converted.)]</p>

デベロッパー サンドボックスを使用して RESTCONF から json または XML に変換する



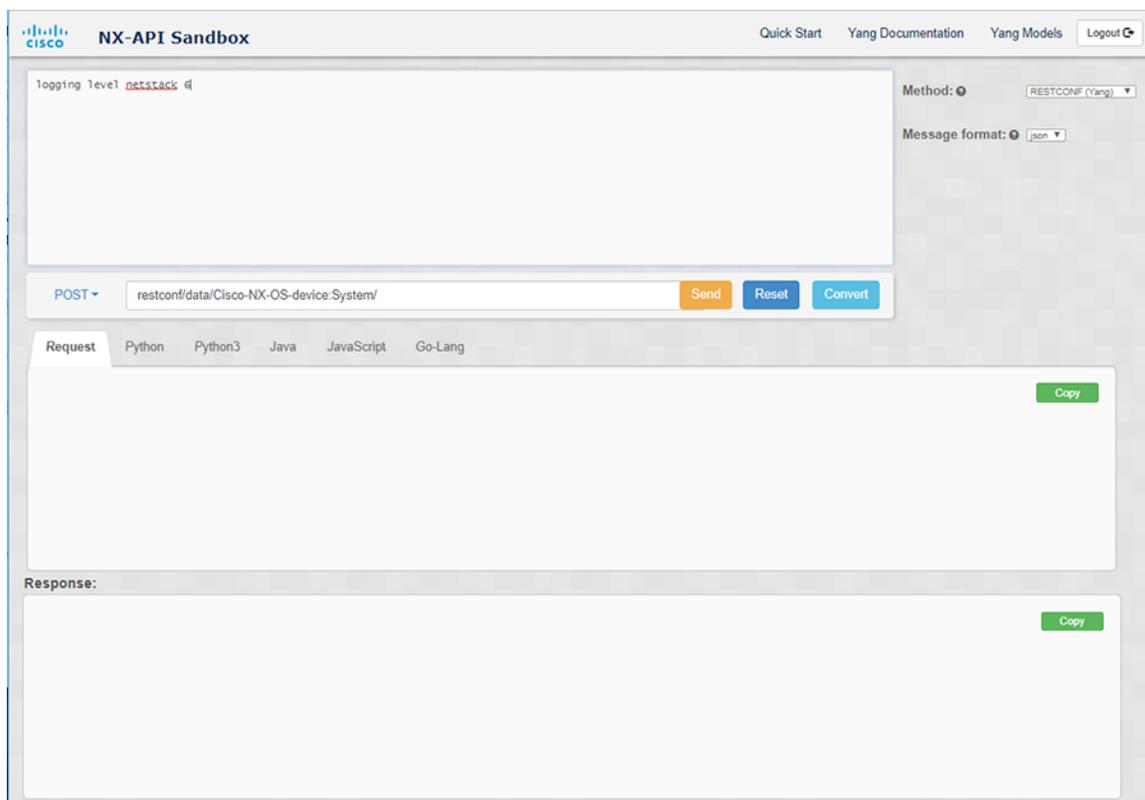
ヒント

- Cisco NX-API Developer Sandbox ウィンドウの右上隅にあるヘルプアイコン (?) をクリックすると、オンラインヘルプを利用できます。
- [サンドボックス] ウィンドウの右上隅にある **Yang Documentation** リンクをクリックして、Model Driven Programmability with Yang ページに移動します。
- [サンドボックス] ウィンドウの右上隅にある **Yang Models** リンクをクリックして、YangModels GitHub サイトにアクセスします。

手順

ステップ1 [メソッド] ドロップダウンリストをクリックし、[RESTCONF (Yang)] を選択します。

例：



ステップ2 [メッセージ形式] をクリックし、**json** または **xml** を選択します。

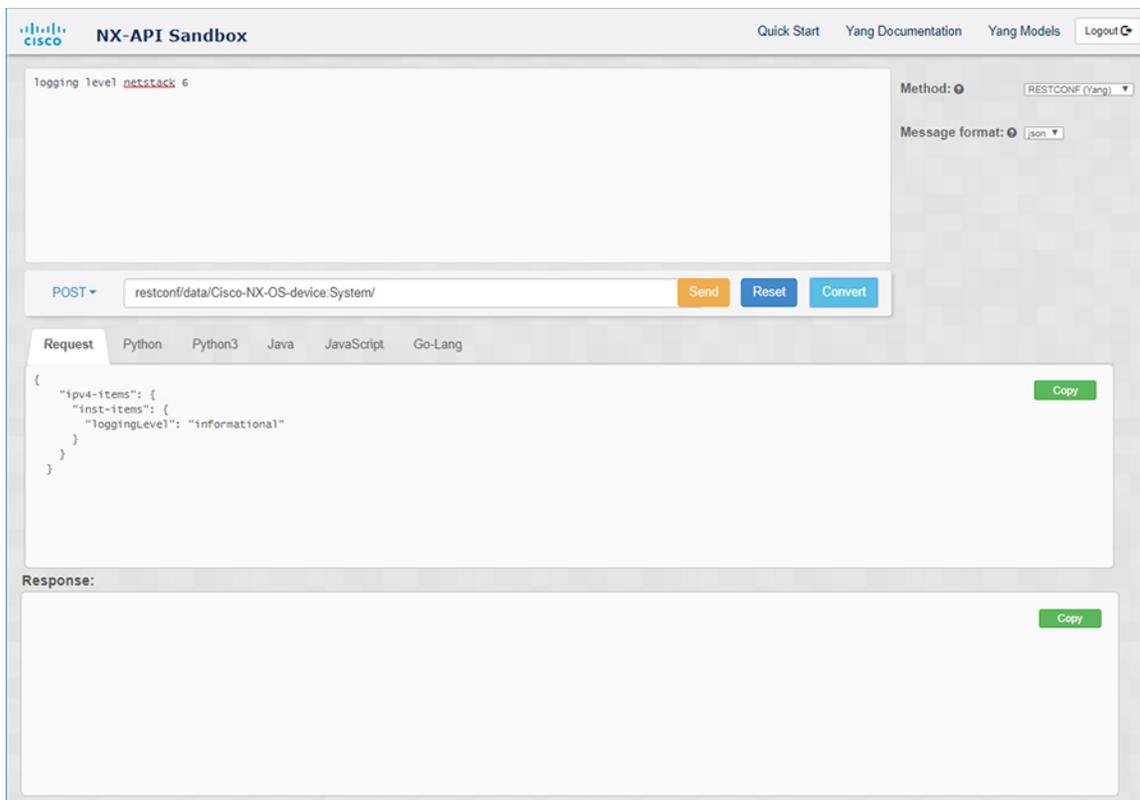
ステップ3 上部ペインのテキスト入力ボックスにコマンドを入力します。

ステップ4 メッセージ形式を選択します。

ステップ5 [変換 (Convert)] をクリックします。

例 :

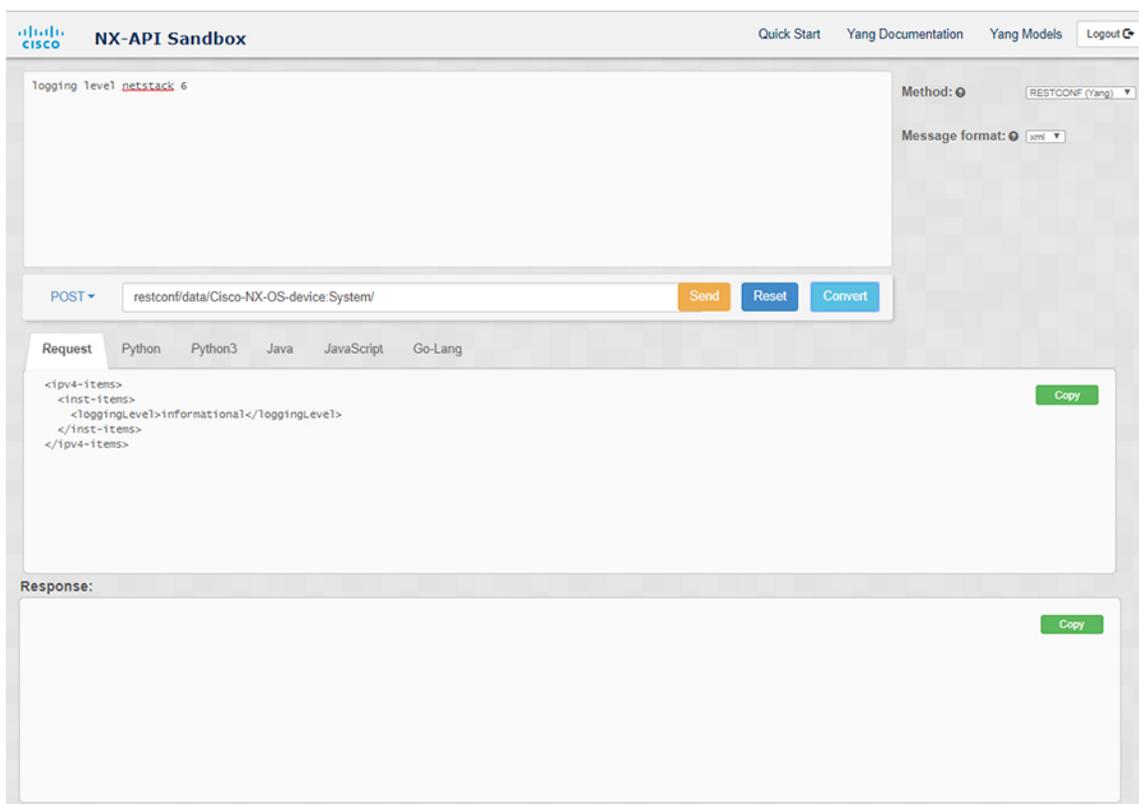
この例では、コマンドはログ レベル **netstack 6** で、メッセージ形式は json です。



例 :

この例では、コマンドはログ レベル **netstack 6** で、メッセージ形式は xml です。

デベロッパーサンドボックスを使用して RESTCONF から json または XML に変換する



(注)

XML または JSON メッセージ形式を使用して、否定された CLI を Yang ペイロードに変換すると、サンドボックスは警告をスローし、**[送信]** オプションを無効にします。表示される警告メッセージは、メッセージの形式によって異なります。

- XML メッセージ形式の場合 — 「これは Netconf ペイロードであり、DELETE 操作用に生成されているため、Restconf では SEND オプションが無効になっています!」
- JSON メッセージ形式の場合 - 「これは、DELETE 操作用に生成される gRPC ペイロードであるため、Restconf では SEND オプションが無効になっています!」

ステップ 6 [リクエスト] ペインの適切なタブをクリックして、リクエストを次の形式に変換することもできます。

- Python
- python3
- Java
- JavaScript
- Go-Lang

(注)

[リクエスト] タブの上の領域にあるドロップダウンメニューから [PATCH] オプションを選択した場合、Java で生成されたスクリプトは機能しません。これは Java の既知の制限であり、予期される動作です。

デベロッパー サンドボックスを使用して **RESTCONF** から **json** または **XML** に変換する

翻訳について

このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。