



NX-API

この章は、次の内容で構成されています。

- [NX-APIについて](#) (1 ページ)
- [NX-API ワークフロー](#) (2 ページ)
- [NX-API のパフォーマンス](#) (2 ページ)
- [NX-API メッセージについて](#) (3 ページ)
- [メッセージ形式](#) (3 ページ)
- [セキュリティ](#) (4 ページ)
- [制限事項](#) (5 ページ)
- [構造化された出力](#) (5 ページ)
- [NX-API CLI の構成](#) (6 ページ)
- [サンプル NX-API スクリプト](#) (10 ページ)
- [構造化出力の例](#) (11 ページ)
- [NX-API 開発者サンドボックス](#) (14 ページ)
- [NX-API リクエスト要素](#) (25 ページ)
- [NX-API 応答要素](#) (31 ページ)
- [デフォルト設定](#) (33 ページ)
- [その他の参考資料](#) (34 ページ)

NX-APIについて

NX-API は、Cisco MDS 9000 シリーズ CLI システムの拡張機能です。

Cisco MDS 9000 NX-API は、CLI コマンドを取得して実行する RPC スタイルの API です。他の Representational State Transfer (REST) API フレームワークとも共通する HTTP または HTTPS プロトコルに基づいて、Cisco MDS スイッチへのプログラムによるアクセスを可能にします。NX-API は、Cisco MDS NX-OS CLI の構成および管理機能を最新の Web ベースの API で提供します。これにより、ユーザは Web ブラウザを使用して Cisco MDS スイッチを制御できます。Python などのプログラミング言語や適切なライブラリと組み合わせると、ストレージネットワークワークの自動化が容易になります。

Cisco MDS NX-API は、特定の **show** コマンドと非インタラクティブな構成コマンドをサポートしています。

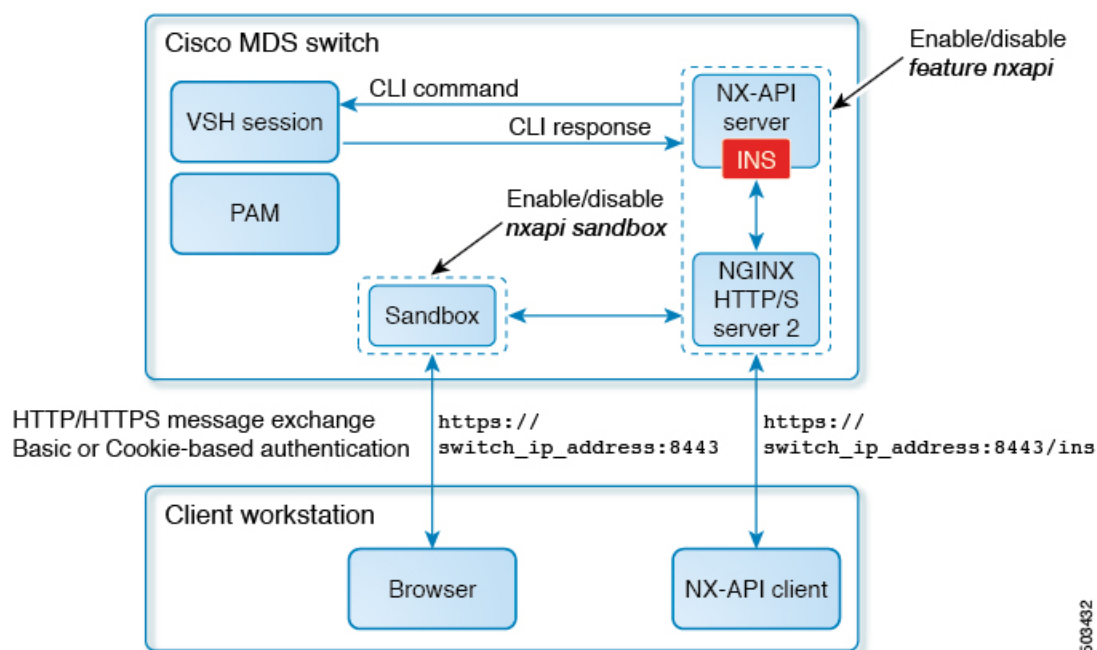


(注) 非インタラクティブコマンドは、ユーザーにキーボードからの入力を求めずに続行するコマンドです。

NX-API ワークフロー

NX-API バックエンドは NGINX HTTP サーバーを使用します。NGINX サーバーは、スイッチ内の外部クライアントと NXAPI サーバー間のインターフェイスとなります。

図 1: NX-API ワークフロー



503-432

NX-API のパフォーマンス

NX-API のスループット パフォーマンスは、次の要因によって変わります。

- **HTTP および HTTPS :** HTTP サーバーでの NX-API のパフォーマンスは、HTTPS サーバーでのパフォーマンスと比較して優れています。これは、HTTPS サーバーには、セキュリティを強化するためにデータを暗号化および復号するオーバーヘッドがあるためです。
- **Cisco MDS スイッチ（メモリおよびプロセスの制限） :** NX-API のパフォーマンスは、より多くのメモリを搭載したデバイスで向上します。

- コマンド出力サイズ：コマンド出力が小さいほど、NX-API のパフォーマンスが向上します。
- **show** コマンドの構造化および非構造化出力：非構造化出力の方が NX-API のパフォーマンスが向上します。構造化された出力をサポートするコマンドは、このマニュアルでは **NX-API 対応コマンド**とも呼ばれます。

NX-API メッセージについて

HTTP Header

ヘッダーを使用すると、クライアントとサーバーは、要求と応答内でコロンで区切られたプロパティと値のペアを使用し、追加情報を渡すことができます。

ここで、NX-API 要求のコンテンツエンコーディングを指定します。サポートされるコンテンツタイプは次のとおりです：

タグ	説明	タイプ	値
content-type	要求のエンコーディングタイプ	文字列	application/json application/json-rpc application/xml

HTTP メソッド

Cisco MDS NX-API は POST メソッドを使用します。

メッセージ本文

メッセージ本文またはペイロードには、HTTP メソッドのデータが含まれます。サポートされているオブジェクトのリストについては、[NX-API 要求要素](#)のセクションを参照してください。

メッセージ応答

メッセージ応答は、HTTP の戻りコードと、メソッドによって返されるデータを含む HTTP 応答本文です。サポートされている要素のリストについては、[NX-API 応答要素](#)のセクションを参照してください。応答コードのリストについては、[NX-API 応答コード](#)の項を参照してください。

メッセージ形式

- サポートされているコマンドの Cisco NX-API 出力は、XML、JSON、および JSON-RPC で表示できます。このメッセージフォーマットは、要求と応答の両方に使用できます。

- XML : XML ペイロードで Cisco MDS NX-OS CLI 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 Cisco MDS NX-OS CLI コマンドを配信するための Cisco NX-API 独自のプロトコル。
- JSON-RPC : JSON ペイロードで Cisco MDS 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。

NX-API は、Cisco NX-OS NETCONF 実装に直接マッピングされません。

セキュリティ

デフォルトでは、Cisco MDS NX-API は HTTP 基本認証を使用します（つまり、すべてのコマンド要求には、HTTP ヘッダーにデバイスのユーザー名とパスワードが含まれている必要があります）。NX-API は、HTTPS を利用してデータを保護および暗号化することもできます。HTTPS 接続では、HTTP 接続よりもセキュリティが強化されます。NX-API は、HTTP 認証方式の代わりにセッションベースの cookie 認証を提供します。

Cisco NX-OS リリース 8.1(x) および 8.2(x) では、HTTPS 経由で NX-API を有効にすると、2048 ビットの SHA-1 自己署名証明書が作成されます。この証明書は 2 年間有効です。期限切れの証明書を使用すると、ブラウザにセキュリティの脆弱性に関する警告が表示されます。このような脆弱性を回避するには、CA 署名付き証明書を使用することをお勧めします。Cisco NX-OS リリース 8.3(1) 以降では、自己署名証明書は 24 時間後に期限切れになります。CA 署名付き証明書を使用することをお勧めします。

CA 証明書の構成については、*Cisco MDS 9000 シリーズセキュリティの設定ガイド*、リリース 8.x の [認証局およびデジタル証明書の構成](#) の章を参照してください。

NX-API は、Cisco MDS スイッチの CLI 認証システムに統合されています。そのため、ユーザは、NX-API を介してポストされた CLI コマンドをスイッチで実行するための適切な権限を持っている必要があります。たとえば、Cisco MDS 9000 スイッチで読み取り専用権限を持つユーザが、NX API を介して構成コマンドを実行することはできません。

NX-API は、スイッチ上のプログラマブル認証モジュール（Programmable Authentication Module、PAM）を使用して認証を行います。PAM の認証数を減らすには、cookie を使用してください。これは PAM の負荷を減らすことにつながります。

NX-API は、ユーザーが最初に認証に成功したときに、セッションベースの cookie、`nxapi_auth` を提供します。`nxapi_auth` cookie は 600 秒（10 分）で期限切れになります。この値は固定されており、構成できません。セッション cookie は、通信中の再認証を回避するために使用されます。セッションベースの cookie が後続の要求に含まれていない場合は、別のセッションベース cookie が必要です。これは完全な認証プロセスによって提供されます。不必要な認証プロセスの使用を避けることで、MDS スイッチのワークロードを軽減できます。

制限事項

- FCIP インターフェイス関連のコマンドの XML 出力はサポートされていません。
- 整合性チェッカー コマンドの XML 出力はサポートされていません。

構造化された出力

NX-OS は、次の構造化された出力フォーマットで、さまざまな **show** コマンドの標準規格出力のリダイレクトをサポートしています。

- XML
- JSON. JSON 出力の上限は 60 MB です。
- JSON Native

Cisco MDS NX-OS CLI で、標準の Cisco MDS NX-OS 出力を JSON、JSON Native、または XML インタープリタに「パイプ接続」すると、これらのフォーマットへの変換が行われます。JSON および XML インタープリタは、Cisco MDS NX-OS ソフトウェアに組み込まれています。たとえば、**show ip access** コマンドを発行し、論理パイプ (|) を使用して、出力フォーマットを指定することができます。Cisco MDS NX-OS コマンド出力が適切に構造化され、そのフォーマットでエンコードされます。この機能により、プログラムによるデータの解析が可能になり、ソフトウェア ストリーミング テレメトリを介したスイッチからのストリーミング データがサポートされます。

さまざまな出力形式を選択する方法の詳細については、[NX-API 開発者サンドボックス（14 ページ）](#) セクションを参照してください。

Cisco MDS NX-OS リリース 8.3(1) から、シスコは JSON Native と呼ばれる JSON の拡張バージョンを実装しています。これは、選択できる新しい CLI オプションです。JSON Native と JSON Pretty Native は、追加のコマンド解釈レイヤーをバイパスすることにより、JSON 出力をより高速かつ効率的に表示します。実際、JSON Native は出力のデータ型を保持します。整数を出力用の文字列に変換するのではなく、整数として表示します。JSON Native を使用することをお勧めします。

JSON について

JavaScript Object Notation（JavaScript オブジェクト表記法、JSON）は、データを人が読み取りやすくするために設計された軽量テキストベースのオープンスタンダードで、XML の代替になります。JSON はもともと JavaScript から設計されましたが、言語に依存しないデータ形式です。コマンド出力では、JSON および JSON Native がサポートされています。

すべての最新プログラミング言語で何らかの方法でサポートされている 2 つの主要なデータ構造は次のとおりです：

- 値の順序付きリスト：多くの場合、配列またはリストと呼ばれます（たとえば Python ではリスト）
- キー/値ペアのコレクション：多くの場合、オブジェクトまたはディクショナリと呼ばれます（たとえば Python ではディクショナリ）

CLI の実行

```
switch# show cdp neighbors | json
{
  "TABLE_cdp_neighbor_brief_info": {
    "ROW_cdp_neighbor_brief_info": {
      "device_id": "SW-DRISHTI-ECBU-L13",
      "interface": "mgmt0",
      "ttl": "168",
      "capability": [
        "switch",
        "IGMP_cnd_filtering"
      ],
      "platform_id": "cisco",
      "port_id": "GigabitEthernet0/7"
    }
  },
  "neigh_count": "1"
}
```

NX-API CLI の構成

Cisco MDS 9000 シリーズ デバイスのコマンド、コマンドタイプ、および出力タイプは、CLI を HTTP/HTTPS POST の本文にエンコードすることにより、Cisco MDS NX-API を使用して入力されます。要求に対する応答は、XML または JSON 出力形式で返されます。

NX-API 応答コードの詳細については、[NX-API 応答コードの表（32 ページ）](#) を参照してください。

MDS スイッチで NX-API を設定すると、次の URL からアクセスできます：

- HTTP - `http://switch_ip_address:port-number/ins`
- HTTPS - `https://switch_ip_address:port-number/ins`

デフォルトの HTTP および HTTPS 設定については、[デフォルト設定（33 ページ）](#) セクションを参照してください。

次の例は、NX-API を構成して有効化する方法を示しています：

1. 管理インターフェイスを介してスイッチにアクセスできることを確認します。

管理インターフェイスを有効にする方法については、*Cisco MDS 9000 シリーズ基礎構成ガイド*の [管理インターフェイスの構成](#)のセクションを参照してください。

2. NX-API 機能を有効にします。

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

3. (オプション) NX-API 機能を無効にします。

```
switch(config)# no feature nxapi
```

4. MDS スイッチで NX-API を設定すると、HTTP/HTTPS ポートを介してアクセスできます：
(オプション) NX-API の HTTP ポートを構成します。

```
switch(config)# nxapi http port 8080
```

このコマンドを無効にするには、コマンドの **no** 形式を使用します。

(オプション) NX-API の HTTPS ポートを構成します。

```
switch(config)# nxapi https port 8443
```

このコマンドを無効にするには、コマンドの **no** 形式を使用します。

5. (オプション) NX-API HTTPS 接続用のアイデンティティ証明書をインストールします。
トラスト ポイントまたは NX-API 証明書のいずれかを使用できます。両方のソースを同時に構成することはできません。

1. NX-API 機能でのみ使用される暗号化されていない秘密キーを使用して証明書をインストールします：

```
switch(config)# nxapi certificate certfile key keyfile
```

2. NX-API 機能でのみ使用される暗号化された秘密キーを使用して証明書をインストールします：

```
switch(config)# nxapi certificate certfile key keyfile password passphrase
```



- (注) 新しい NX-API 証明書をインストールすると、NX-API サーバーがリセットされます。NX-API を使用してホストから証明書をインストールすると、スクリプトが失敗する可能性があります。

トラストポイントの構成については、[Cisco MDS 9000 シリーズセキュリティ構成ガイド、リリース 8.x の認証局およびデジタル証明書の構成](#)の章を参照してください。

- *certfile* は、プライバシー強化メール (PEM) フォーマットによる、このスイッチの署名付き証明書です。PEM フォーマットは、1993 年の一連の IETF 標準に基づいて、暗号化 RSA キー、証明書、およびその他のデータを保存および送信するための標準ファイルフォーマットです。
- *keyfile* は、このスイッチの PEM フォーマットの秘密キーです。キーが暗号化されている場合は、**password** オプションも指定する必要があります。
- *passphrase* は、秘密キーの暗号化で使用するパスワードです。
- *label* は、すでに構成されている暗号化トラストポイントの名前です。



- (注)
- **nxapi certificate key** コマンドを使用してインストールされた証明書とキーは、スイッチ上の他の暗号機能と共有されません。
 - **nxapi certificate key** コマンドで使用する **password** オプションは、Cisco MDS NX-OS リリース 8.5(1) でのみ使用できます。

6. (オプション) 必要に応じて、NX-API HTTPS 接続に弱い SSL 暗号を許可します。この場合、SSL 接続のセキュリティが低下します。ただし、これは、古いデバイスがスイッチと通信するために必要な場合があります。

```
switch(config)# nxapi ssl ciphers weak
```

7. (オプション) 必要に応じて、NX-API HTTPS 接続の SSL トランスポートを構成します。デフォルト以外の古いトランスポートを有効にすると、SSL 接続のセキュリティが低下します。ただし、これは、古いデバイスがスイッチと通信するために必要な場合があります。LDAP クライアントとサーバー間の SSL を設定するには、[LDAP サーバーの SSL トランスポートの構成](#)を参照してください。

```
• switch(config)# nxapi ssl protocols TLSv1.1 TLSv1.2 TLSv1.3
```

```
• switch(config)# nxapi ssl protocols TLSv1.3
```

NX-API で使用するためのアイデンティティ証明書の準備

NX-API のアイデンティティ証明書は、**nxapi certificate** コマンドを使用してインポートする前に、作成しておく必要があります。証明書は、スイッチアイデンティティ証明書のみで構成する必要があります。すべての CA 証明書と中間認証局の証明書を削除する必要があります。秘密キーを削除する必要があります。また、合計サイズが 4096 バイト未満である必要があります。

スイッチアイデンティティ証明書がトラストポイントのスイッチ暗号化インフラストラクチャにすでにインストールされている場合は、これをエクスポートして再フォーマットし、秘密キーを抽出して NX-API で使用できます。スイッチアイデンティティ証明書がまだインストールされていない場合は、CA によって作成される必要があります。

証明書の構成については、*Cisco MDS 9000 シリーズ セキュリティの設定ガイド*、リリース 8.x の[証明書認証およびデジタル証明書の構成](#)の章を参照してください。



- (注) NX-API で使用する既存のアイデンティティ証明書を準備するためのツールは、スイッチでは使用できません。これは、OpenSSL がインストールされているホストなどの別のデバイスで実行する必要があります。

1. (オプション) スイッチアイデンティティ証明書がすでにスイッチにインストールされている場合は、次のコマンドを使用して PKCS12 フォーマットでエクスポートします:

```
switch(config)# crypto ca export trustpoint_name pkcs12 mytpexport.pkcs12 my_passphrase
```

2. OpenSSL がインストールされているホストにファイルをアップロードします：

```
switch# copy mytpexport.pkcs12 sftp://10.10.2.2
```

3. アイデンティティ証明書を抽出します：

```
host$ openssl pkcs12 -in mytpexport.pkcs12 -nokeys -clcerts -out idcert.pem
Enter Import Password: my_passphrase
host$
```

4. 暗号化されていない秘密キーを抽出します：

```
host$ openssl pkcs12 -in mytpexport.pkcs12 -nocerts -nodes | openssl rsa -out
unencryptedprivkey.pem
Enter Import Password:
writing RSA key
host$
```

5. スイッチ上のブートフラッシュに 2 つのファイルをダウンロードします：

```
switch# copy sftp://10.10.2.2/idecert.pem bootflash:
switch# copy sftp://10.10.2.2/unencryptedprivkey.pem bootflash:
```

nxapi certificate コマンドを使用してファイルをインポートする準備ができました。

cURL での NX-API の使用

ホスト上の **show.version.json** ファイルの内容を調べてみましょう。

```
linux$ cat show.version.json
[{ "jsonrpc": "2.0", "method": "cli", "params": { "cmd": "show version", "version": 1
}, "id": 1 }]
EOF
```

ホストで cURL を使用してスイッチを認証し、目的の POST 要求を送信します。

```
linux$ curl -v -u admin:cisco -H "Content-Type: application/json-rpc" -H "Cache-Control:
no-cache" -d @show.version.json -X POST http://10.10.2.2:80/ins
```

```
Note: Unnecessary use of -X or --request, POST is already inferred.
* Trying 10.10.2.2:80...
* Connected to 10.10.2.2:80 (10.10.2.2:80) port 80 (#0)
* Server auth using Basic with user 'admin'
> POST /ins HTTP/1.1
> Host: 10.10.2.2:80
> Authorization: Basic YWRtaW46bmJ2XzEyMzQ1
> User-Agent: curl/7.70.0
> Accept: */*
> Content-Type: application/json-rpc
> Cache-Control: no-cache
> Content-Length: 99
>
* upload completely sent off: 99 out of 99 bytes
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.7.10
< Date: Mon, 14 Jun 1976 13:28:43 GMT
< Content-Type: application/json-rpc; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< Set-Cookie: nxapi_auth=dzqnf:1fNa+E8KGq0ZZM6TRZTFKTWejBg=; Secure; HttpOnly;
< X-Frame-Options: SAMEORIGIN
```

```

< X-XSS-Protection: 1; mode=block
< X-Content-Type-Options: nosniff
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< Content-Security-Policy: block-all-mixed-content; base-uri 'self'; default-src 'self';
  script-src 'self'; style-src 'self'; img-src 'self'; connect-src 'self'; font-src
'self'; object-src 'none'; media-src 'self'; form-action 'self'; frame-ancestors 'self';
<
{
  "jsonrpc": "2.0",
  "result": {
    "header_str": "Cisco Nexus Operating System (NX-OS) Software\nTAC
support: http://www.cisco.com/tac\nDocuments:
http://www.cisco.com/en/US/products/ps9372/tsd_products_support_series_home.html\nCopyright
(c) 2002-2020, Cisco Systems, Inc. All rights reserved.\nThe copyrights to certain works
contained herein are owned by\nother third parties and are used and distributed under
license.\nSome parts of this software are covered under the GNU Public\nLicense. A copy
of the license is available at\nhttp://www.gnu.org/licenses/gpl.html.\n",
    "bios_ver_str": "2.1.17",
    "loader_ver_str": "N/A",
    "kickstart_ver_str": "8.4(1)SK(0) [build 8.4(1)SK(0.160)] [gdb]",
    "sys_ver_str": "8.4(1)SK(0) [build 8.4(1)SK(0.160)] [gdb]",
    "bios_cmpl_time": "01/08/14",
    "kick_file_name": "bootflash:///kick-sky160",
    "kick_cmpl_time": " 12/20/2020 12:00:00",
    "kick_tmstamp": "09/08/2020 09:42:15",
    "isan_file_name": "bootflash:///sky-sep14-02",
    "isan_cmpl_time": " 12/20/2020 12:00:00",
    "isan_tmstamp": "09/14/2020 05:56:35",
    "chassis_id": "MDS 9250i 40 FC 2 IPS 8 FCoE (2 RU) Chassis",
    "module_id": "40FC+8FCoE+2IPS Supervisor",
    "cpu_name": "Motorola, e500v2",
    "memory": 4088480,
    "mem_type": "kB",
    "proc_board_id": "JAF1852AAFC",
    "host_name": "host",
    "bootflash_size": 4001760,
    "kern_uptime_days": 0,
    "kern_uptime_hrs": 1,
    "kern_uptime_mins": 25,
    "kern_uptime_secs": 13,
    "rr_usecs": 715180,
    "rr_ctime": "Mon Jun 14 12:02:47 1976",
    "rr_reason": "Reset Requested by CLI command reload",
    "rr_sys_ver": "8.4(1)SK(0.160)",
    "rr_service": "",
    "manufacturer": "Cisco Systems, Inc."
  },
  "id": 1,
  "cmd": "show version"
}
* Connection #0 to host 10.197.155.246 left intact

```

サンプル NX-API スクリプト

ユーザーはNX-APIでスクリプトを使用する方法を示すサンプルスクリプトにアクセスできます。サンプルスクリプトにアクセスするには、次のリンクをクリックして、必要なソフトウェアリリースに対応するディレクトリを選択します：<https://github.com/datacenter/nxos/tree/master/nxapi/samples>

構造化出力の例

このセクションでは、XML、JSON および JSON ネイティブ出力フォーマットとして表示される Cisco MDS NX-OS コマンドのいくつかの例を取り上げます。

特定の **show** コマンドが NX-API 対応かどうかを確認するには、スイッチ上で、**|xml** とともにコマンドを入力します：

command | xml

コマンドが NX-API 対応（構造化出力をサポート）の場合、結果の出力は XML フォーマットになります：

```
switch# show device-alias merge status | xml

<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns="http://www.cisco.com/nxos:8.4.1.SK.0.:ddas"
xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0">
<nf:data>
<show>
<device-alias>
<merge>
<status>
<__readonly__>
<result>Success</result>
<reason>None</reason>
</__readonly__>
</status>
</merge>
</device-alias>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

コマンドが NX-API 対応でない場合、結果の出力に次のエラーが表示されます：

```
switch# show logging logfile | xml

Error: This command does not support XML output.
```

次に、**show version** コマンドを XML フォーマットで表示する例を示します：

```
switch(config)# show version | xml

<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns="http://www.cisco.com/nxos:8.4.2.:sysmgrcli"
xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0">
<nf:data>
<show>
<version>
<__readonly__>
<header_str>Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Documents: http://www.cisco.com/en/US/products/ps9372/tsd_products_support_series_home.html
Copyright (c) 2002-2020, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained in this software are
owned by other third parties and used and distributed under
license. Certain components of this software are licensed under
```

```

the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php
</header_str>
  <bios_ver_str>3.7.0</bios_ver_str>
  <kickstart_ver_str>8.4(2) [build 8.4(2.191)] [gdb]</kickstart_ver_str>
  <sys_ver_str>8.4(2) [build 8.4(2.191)] [gdb]</sys_ver_str>
  <bios_cmpl_time>04/01/2019</bios_cmpl_time>
  <kick_file_name>bootflash:///m9700-sf3ek9-kickstart-mzg.8.4.2.191.bin</kick_file_name>

  <kick_cmpl_time> 2/5/2020 12:00:00</kick_cmpl_time>
  <kick_tmstamp>01/08/2020 18:27:03</kick_tmstamp>
  <isan_file_name>bootflash:///m9700-sf3ek9-mzg.8.4.2.191.bin</isan_file_name>
  <isan_cmpl_time> 2/5/2020 12:00:00</isan_cmpl_time>
  <isan_tmstamp>01/14/2020 05:36:15</isan_tmstamp>
  <chassis_id>MDS 9706 (6 Slot) Chassis</chassis_id>
  <module_id>Supervisor Module-3</module_id>
  <cpu_name>Intel(R) Xeon(R) CPU C5528 @ 2.13GHz</cpu_name>
  <memory>8167228</memory>
  <mem_type>kB</mem_type>
  <proc_board_id>JAE19220AQJ</proc_board_id>
  <host_name>abc</host_name>
  <bootflash_size>3915776</bootflash_size>
  <slot0_size>0</slot0_size>
  <kern_uptm_days>19</kern_uptm_days>
  <kern_uptm_hrs>23</kern_uptm_hrs>
  <kern_uptm_mins>16</kern_uptm_mins>
  <kern_uptm_secs>11</kern_uptm_secs>
  <rr_usecs>768558</rr_usecs>
  <rr_ctime>Tue Jan 14 05:58:26 2020</rr_ctime>
  <rr_reason>Reset Requested by CLI command reload</rr_reason>
  <rr_sys_ver>8.4(2.171)</rr_sys_ver>
  <rr_service></rr_service>
  <manufacturer>Cisco Systems, Inc.</manufacturer>
</__readonly__>
</version>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>

```

次に、**show version** を JSON フォーマットで表示する例を示します：

```

switch(config)# show version | json

{
  "header_str": "Cisco Nexus Operating System (NX-OS) Software\nTAC support:
http://www.cisco.com/tac\nDocuments: http://www.cisco.c
om/en/US/products/ps9372/tsd_products_support_series_home.html\nCopyright (c) 2002-2020,
Cisco Systems, Inc. All rights reserved.\nT
he copyrights to certain works contained in this software are\nowned by other third
parties and used and distributed under\nlicense.
Certain components of this software are licensed under\nthe GNU General Public License
(GPL) version 2.0 or the GNU\nLesser General
Public License (LGPL) Version 2.1. A copy of each\nsuch license is available
at\nhttp://www.opensource.org/licenses/gpl-2.0.php and
\nhttp://www.opensource.org/licenses/lgpl-2.1.php",
  "bios_ver_str": "3.7.0",
  "kickstart_ver_str": "8.4(2) [build 8.4(2.191)] [gdb]",
  "sys_ver_str": "8.4(2) [build 8.4(2.191)] [gdb]",
  "bios_cmpl_time": "04/01/2019",
  "kick_file_name": "bootflash:///m9700-sf3ek9-kickstart-mzg.8.4.2.191.bin",

```

```

"kick_cmpl_time": "2/5/2020 12:00:00",
"kick_tmstamp": "01/08/2020 18:27:03",
"isana_file_name": "bootflash:///m9700-sf3ek9-mzg.8.4.2.191.bin",
"isana_cmpl_time": "2/5/2020 12:00:00",
"isana_tmstamp": "01/14/2020 05:36:15",
"chassis_id": "MDS 9706 (6 Slot) Chassis",
"module_id": "Supervisor Module-3",
"cpu_name": "Intel(R) Xeon(R) CPU C5528 @ 2.13GHz",
"memory": 8167228,
"mem_type": "kB",
"proc_board_id": "JAE19220AQJ",
"host_name": "abc",
"bootflash_size": 3915776,
"slot0_size": 0,
"kern_uptime_days": 19,
"kern_uptime_hrs": 23,
"kern_uptime_mins": 16,
"kern_uptime_secs": 22,
"rr_usecs": 768558,
"rr_ctime": "Tue Jan 14 05:58:26 2020",
"rr_reason": "Reset Requested by CLI command reload",
"rr_sys_ver": "8.4(2.171)",
"rr_service": null,
"manufacturer": "Cisco Systems, Inc."
}

```

次に、**show version** を JSON ネイティブ フォーマットで表示する例を示します：

```

switch(config)# show version | json native

{
  "header_str": "Cisco Nexus Operating System (NX-OS) Software\nTAC support: http://www.cisco.com/tac\nDocuments: http://www.cisco.com/en/US/products/ps9372/tsd_products_support_series_home.html\nCopyright (c) 2002-2020, Cisco Systems, Inc. All rights reserved.\nThe copyrights to certain works contained herein are owned by\nother third parties and are used and distributed under license.\nSome parts of this software are covered under the GNU Public\nLicense. A copy of the license is available at\nhttp://www.gnu.org/licenses/gpl.html.\n",
  "bios_ver_str": "2.1.18",
  "loader_ver_str": "N/A",
  "kickstart_ver_str": "8.4(2a)",
  "sys_ver_str": "8.4(2a)",
  "bios_cmpl_time": "04/06/20",
  "kick_file_name": "bootflash:///m9100-s5ek9-kickstart-mz.8.4.2a.bin",
  "kick_cmpl_time": " 7/11/2020 12:00:00",
  "kick_tmstamp": "06/20/2020 20:50:09",
  "isana_file_name": "bootflash:///m9100-s5ek9-mz.8.4.2a.bin",
  "isana_cmpl_time": " 7/11/2020 12:00:00",
  "isana_tmstamp": "06/20/2020 22:05:47",
  "chassis_id": "MDS 9148S 16G 48 FC (1 Slot) Chassis",
  "module_id": "2/4/8/16 Gbps FC/Supervisor",
  "cpu_name": "Motorola, e500v2",
  "memory": 4088620,
  "mem_type": "kB",
  "proc_board_id": "JAF1751BGPS",
  "host_name": "sw109-Mini",
  "bootflash_size": 4001760,
  "kern_uptime_days": 7,
  "kern_uptime_hrs": 1,
  "kern_uptime_mins": 13,
  "kern_uptime_secs": 0,
  "rr_usecs": 362070,
  "rr_ctime": "Mon Sep 28 07:43:36 2020",

```

NX-API 開発者サンドボックス

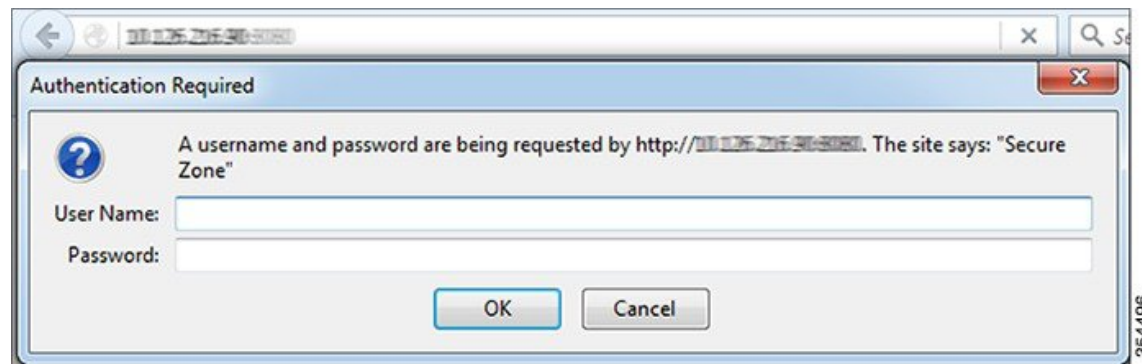


(注) NX-API 開発者サンドボックスを使用する場合は、Firefox リリース 24.0 以降を使用することを推奨します。NX-API 開発者サンドボックスの **[コピー (Copy)]** ボタンと **[Python]** ボタンが機能するには、ブラウザに最新の Adobe Flash プレーヤーがインストールされている必要があります。

1. ブラウザを開き、そのアドレスバーに、HTTP の場合は `http://switch_ip_address:port-number`、HTTPS の場合は `https://switch_ip_address:port-number` と入力します。

[NX-API 開発者サンドボックス認証 (NX-API Developer Sandbox Authentication)] ウィンドウが表示されます。

図 2: NX-API 開発者サンドボックス認証



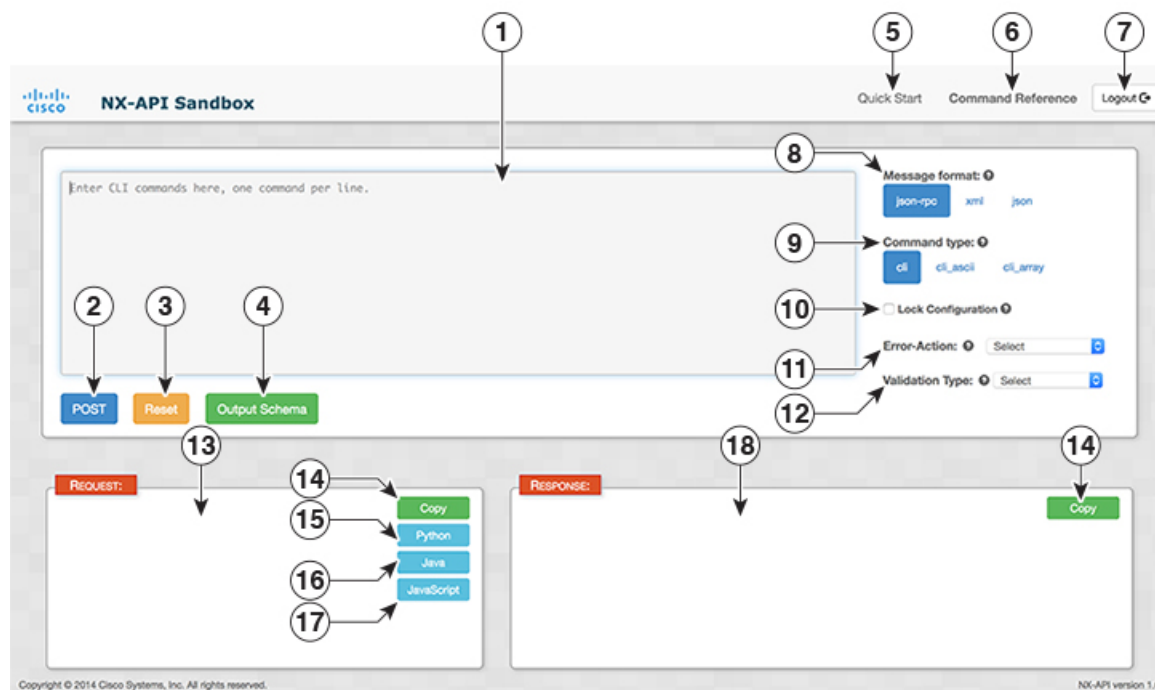
2. 適切なクレデンシャルでログインします。

[NX-API 開発者サンドボックス (NX-API Developer Sandbox)] ウィンドウが表示されます。

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 ペイロードを同等の CLI に変換します。

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

図 3: NX-API 開発者サンドボックス



1	コマンド入力：コマンドを入力できます。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。	10	[構成のロック (Lock Configuration)]：構成を排他的にロックします。これにより、他の管理エージェントは構成を変更できなくなります。
2	POST：特定のコマンドの出力を生成します。	11	[エラーアクション (Error Action)]：エラーアクションのオプションを指定します。 • Stop-on-error：最初に失敗した CLI で停止します。 • Continue-on-error：他の CLI を無視して続行します。 • Rollback-on-error：システムを以前の状態にロールバックします。 (注) rollback-on-error オプションは、Cisco MDS NX-OS リリース 9.2(2) から削除されました。

3	[リセット (Reset)] : コマンドおよび対応する出力をクリアします。	12	[検証タイプ (Validation Type)] : 検証設定を指定します。 • Validate-Only : 構成を検証しますが、構成は適用しません。 • Validate-and-Set : 設定を検証し、検証が成功した場合はスイッチに設定を適用します。
4	[出力スキーマ (Output Schema)] : コマンドペインに入力されたコマンドのコマンドスキーマを表示します。	13	[要求 (REQUEST)] : 入力されたコマンドの出力を、選択したメッセージフォーマットで表示します。
5	[クイックスタート (Quick Start)] : Cisco MDS NX-API のオンラインヘルプを表示します。	14	[コピー (Copy)] : [要求 (REQUEST)] または [応答 (RESPONSE)] エリアに入力されたデータをコピーします。
6	[コマンドリファレンス (Command Reference)] : [コマンドリファレンス (Command Reference)] ペインを表示します。 [コマンドリファレンス (Command Reference)] ペインには、 [コマンドの表示 (Show commands)] ペインで選択したコマンドのコマンドスキーマの詳細が表示されます。 (注) Cisco MDS NX-OS リリース 8.4(1) 以降でサポートされます。	15	Python
7	[ログアウト (Logout)] : NX-API サンドボックスからユーザーをログアウトします。	16	Java (注) Cisco MDS NX-OS リリース 8.4(1) 以降でサポートされます。

8	[メッセージフォーマット (Message format)] : コマンド出力を表示するさまざまなメッセージフォーマットを指定します。	17	Javascript (注) Cisco MDS NX-OS リリース 8.4(1) 以降でサポートされます。
9	コマンド タイプ • cli — コマンドを表示または構成します。 • cli_ascii — コマンドをフォーマットせずに表示または構成します。 • cli_array — CLI の show コマンド構造化された出力が必要な CLI show コマンド。show コマンド専用。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。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。 (注) cli_array コマンドタイプは、Cisco MDS NX-OS リリース 8.4(1) からサポートされています。	18	[応答 (RESPONSE)] : コマンド入力領域に入力されたコマンドの API 応答を表示します。

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 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。

NX-API 開発者サンドボックスを使用してコマンドの出力を生成するには、次の手順を実行します：

1. コマンド出力を表示するメッセージフォーマットタイプ (**json-rpc**、**xml**、**json**) をクリックします。(デフォルトでは、**json-rpc** が選択されています)。
2. 入力したコマンドタイプをクリックします。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。(デフォルトでは、**cli** が選択されています)。

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。



- (注) **xml** メッセージフォーマットを選択した場合は、**cli_show** および **cli_show_ascii** コマンドタイプのチャンクモードを有効にできます。**[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オンにして、大きな **show** コマンド出力がチャンクされるようにします。出力の次のチャンクを表示するには、**[応答 (RESPONSE)]** エリアの **<sid>** および **</sid>** タグの間に記載されているセッションID (SID) をコピーし、**[チャンクモードを有効にする (Enable Chunk mode)]** チェックボックスの下に **[SID]** ボックスに貼り付けます。

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。
4. 入力したコマンドが、選択したメッセージフォーマットで **[要求 (REQUEST)]** エリアに表示されます。

リクエストペインにも一連のタブがあります。各タブは異なる言語を表しており、**Python**、**Java**、**JavaScript** に対応しています。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。

- **[要求 (REQUEST)]** エリアに入力されたデータをコピーするには、**[コピー (Copy)]** をクリックします。
- 入力したコマンドの Python コードを生成するには、**[Python]** をクリックします。



- (注) **xml** メッセージフォーマットは、**[Python]** ボタンをサポートしていません。

5. **[POST]** をクリックして、コマンドの出力を生成します。

コマンドの出力が **[応答 (RESPONSE)]** エリアに表示されます。

- **[応答 (RESPONSE)]** エリアに入力されたデータをコピーするには、**[コピー (Copy)]** をクリックします。

コマンドとそれに対応する出力をクリアし、ページをリセットするには、[リセット (Reset)] をクリックします。

6. NX-API でサポートされている show コマンドのリストを表示するには、[コマンドリファレンス (Command Reference)] をクリックします。

[コマンドスキーマ (Command Schema)] ペインには、[コマンドの表示 (Show commands)] ペインで選択したコマンドの詳細が表示されます。



(注) 現在、[コマンドリファレンス (Command Reference)] タブは show コマンドのみをサポートしています。

図 4: NX-API Show コマンドリファレンス

1 **Show commands**
Please click the command to see the schema

- show system login
- show system login failures
- show system reset-reason
- show system poap
- show system error-id list
- show system exception-info
- show system auto-upgrade epid status
- show system vqi allocations
- show system vian reserved
- show system cores
- show system pas shrink status details
- show system pas shrink status
- show system redundancy status
- show system redundancy ha status
- show system standby manual-boot
- show system uptime
- show system auto-collect tech-support
- show system inband queuing status
- show system inband queuing statistics
- show system resources
- show system default zone
- show feature
- show feature-set
- show hardware
- show hardware capacity module
- show hardware capacity power
- show hardware capacity eobc
- show hardware capacity fabric-utilization

2 **Command Schema**

show system login.

Field	Data Type	Description
acc_list	string	Applied ACL's
attempts	integer	Number of login failures
block_for	integer	Login disabled for time
fail_count	integer	Login failure count
switch_mode	string	Mode of operation
time	integer	Time remaining to re-enable login
within	integer	Number of login failures within time

1	show コマンド：サポートされている show コマンドのリストを表示します。
2	[コマンドスキーマ (Command Schema)] : [コマンドの表示 (Show commands)] ペインで選択したコマンドの NX-API スキーマ (キーワードおよび説明) を表示します。

例：NX-API ステータスの表示

次に、NX-API ステータス応答をさまざまな出力フォーマットで表示する例を示します。

XML 形式

show nxapi

要求:

```
<?xml version="1.0"?>
<ins_api>
  <version>1.2</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>show nxapi</input>
  <output_format>xml</output_format>
</ins_api>
```

応答:

```
<ins_api>
  <type>cli_show</type>
  <version>1.2</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body>
        <nxapi_status>Enabled</nxapi_status>
        <sandbox_status>Enabled</sandbox_status>
        <http_port>8080</http_port>
      </body>
      <input>show nxapi</input>
      <msg>Success</msg>
      <code>200</code>
    </output>
  </outputs>
</ins_api>
```

[JSON 形式 (JSON Format)]

show nxapi

要求:

```
{
  "ins_api": {
    "version": "1.2",
    "type": "cli_show",
    "chunk": "0",
    "sid": "1",
    "input": "show nxapi",
    "output_format": "json"
  }
}
```

応答:

```
{
  "ins_api": {
    "type": "cli_show",
    "version": "1.2",
```

```

    "sid": "eoc",
    "outputs": {
      "output": {
        "input": "show nxapi",
        "msg": "Success",
        "code": "200",
        "body": {
          "nxapi_status": "Enabled",
          "sandbox_status": "Enabled",
          "http_port": "8080"
        }
      }
    }
  }
}

```

JSON-RPC フォーマット

show nxapi

要求:

```

[
  {
    "jsonrpc": "2.0",
    "method": "cli",
    "params": {
      "cmd": "show nxapi",
      "version": 1.2
    },
    "id": 1
  }
]

```

応答:

```

{
  "jsonrpc": "2.0",
  "result": {
    "body": {
      "nxapi_status": "Enabled",
      "sandbox_status": "Enabled",
      "http_port": "8080"
    }
  },
  "id": 1
}

```

例: VSAN から VLAN へのマッピングの構成

次に、グローバル構成モード (**cli_conf**) で、VSAN から VLAN へのマッピングを構成する例を示します。

```

vlan 3
fcoe vsan 3
vsan database
vsan 3
vsan 3 interface vfc1/8

```

要求:

```
<?xml version="1.0"?>
<ins_api>
  <version>1.2</version>
  <type>cli_conf</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>vlan 3 ;fcoe vsan 3 ;vsan database ;vsan 3 ;vsan 3 interface vfc1/8</input>
  <output_format>xml</output_format>
</ins_api>
```

応答 :

```
<?xml version="1.0"?>
<ins_api>
  <type>cli_conf</type>
  <version>1.2</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body/>
      <input>vlan 3</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>fcoe vsan 3</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>vsan database</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>vsan 3</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>vsan 3 interface vfc1/8</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
  </outputs>
</ins_api>
```

例 : ゾーンとゾーンセットの構成

次に、グローバル構成モード (**cli_conf**) でゾーンを構成する例を示します。

```
zone name zone2 vsan 1
member pwnn 10:00:00:23:45:67:89:ab
member pwnn 10:00:00:23:45:67:89:cd
```

要求:

```
<?xml version="1.0"?>
<ins_api>
  <version>1.2</version>
```

```

<type>cli_conf</type>
<chunk>0</chunk>
<sid>sid</sid>
<input>zone name zone2 vsan 1 ;member pwnn 10:00:00:23:45:67:89:ab ;member pwnn
10:00:00:23:45:67:89:cd</input>
<output_format>xml</output_format>
</ins_api>

```

応答 :

```

<?xml version="1.0"?>
<ins_api>
  <type>cli_conf</type>
  <version>1.2</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body/>
      <input>zone name zone2 vsan 1</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>member pwnn 10:00:00:23:45:67:89:ab</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>member pwnn 10:00:00:23:45:67:89:cd</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
  </outputs>
</ins_api>

```

次に、グローバル構成モード (**cli_conf**) でゾーンセットを構成する例を示します。

```

zoneset name Zoneset1 vsan 1
member zone2
zoneset activate name Zoneset1 vsan 1

```

要求:

```

<?xml version="1.0"?>
<ins_api>
  <version>1.2</version>
  <type>cli_conf</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>zoneset name Zoneset1 vsan 1 ;member zone2 ;zoneset activate name Zoneset1 vsan
1</input>
  <output_format>xml</output_format>
</ins_api>

```

応答 :

```

<?xml version="1.0"?>
<ins_api>
  <type>cli_conf</type>
  <version>1.2</version>
  <sid>eoc</sid>
  <outputs>
    <output>

```

```

<body/>
<input>zoneset name Zoneset1 vsan 1</input>
<code>200</code>
<msg>Success</msg>
</output>
<output>
  <body/>
  <input>member zone2</input>
  <code>200</code>
  <msg>Success</msg>
</output>
<output>
  <body>Zoneset activation initiated. check zone status
</body>
<input>zoneset activate name Zoneset1 vsan 1</input>
<code>200</code>
<msg>Success</msg>
</output>
</outputs>
</ins_api>

```

show コマンドが NX-API 対応でない場合でも、JSON および XML エンコード要求の場合は **Command type** 要素を **cli_show_ascii** に、JSON-RPC エンコード要求の場合は **show_ascii** に設定することで、出力にアクセスできます。コマンド出力は、単一のフラット文字列として応答本文で返されます。

次の図に、NX-API 開発者サンドボックスでの NX-API 対応ではない **show** コマンドの出力例を示します。

The screenshot displays the NX-API Developer Sandbox interface. At the top, the input field contains the command `show logging logfile`. To the right, configuration options are set: **Message format** is `json-rpc`, **Command type** is `cli` (with `cli_ascii` selected), and **Lock Configuration** is unchecked. Below these are buttons for **POST**, **Reset**, and **Output Schema**. The interface is split into two main sections: **REQUEST** and **RESPONSE**. The **REQUEST** section shows a JSON-RPC call: `{ "jsonrpc": "2.0", "method": "cli_show_ascii", "params": { "cmd": "show logging logfile", "version": "1.2" }, "id": 1 }`. The **RESPONSE** section shows the corresponding JSON-RPC response: `{ "jsonrpc": "2.0", "result": { "msg": "2020 Jul 17 13:16:55 %SYSLOG-2-SYSTEM_MSG : Syslogs wont be logged into logflash until logflash is enabled", "id": 1, "cmd": "show logging logfile" } }`. Copy buttons are available for both request and response JSON.

NX-API リクエスト要素

NX-API 要求要素は、XML フォーマット、JSON フォーマット、または JSON-RPC フォーマットでスイッチに送信されます。リクエストの HTTP ヘッダーは、リクエストのコンテンツタイプを識別する必要があります。



(注) ロックを保持しているセッションが何らかの理由で終了した場合、ロックはシステムによって解放されます。ロックを取得したセッションは、必要な構成のみを実行できます。

表 1: XML または JSON 形式の NX-API 要求要素

NX-API リクエスト要素	説明
version	NX-API バージョンを指定します。

NX-API リクエスト要素	説明
<i>type</i>	

NX-API リクエスト要素	説明
	実行するコマンドのタイプを指定します。 次のタイプのコマンドがサポートされています。 • cli : CLI 構成コマンド 構造化された出力が必要な CLI show コマンド。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。 • cli_array : CLI show コマンド 構造化された出力が必要な CLI show コマンド。show コマンド専用。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。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。 • cli_ascii : CLI 構成コマンド ASCII 出力が必要な CLI show コマンド。これは、ASCII 出力を解析する既存のスクリプトと一致します。ユーザーは、最小限の変更で既存のスクリプトを使用できます。 • cli_show : 構造化された出力が必要な CLI show コマンド。コマンドが XML 出力をサポートしていない場合は、エラーメッセージが返されます。 • cli_show_array : CLI 構成コマンド。 構造化された出力が必要な CLI show コマンド。show コマンド専用。cli_show に似ていますが、cli_show_array 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。 • cli_show_ascii : ASCII 出力が必要な CLI show コマンド。これは、ASCII 出力を解析する既存のスクリプトと一致します。ユーザーは、最小限の変更で既存のスクリプトを使用できます。 • cli_conf : CLI 構成コマンド。 (注) • 各コマンドは、現在のユーザーの権限でのみ実行可能です。 • 最大 10 の連続する show コマンドがサポートされています。show コマンドの数が 10 を超える場合、11 番目以降のコマンドは無視されます。

NX-API リクエスト要素	説明
	対話型コマンドはサポートされていません。
チャンク	一部の show コマンドは、大量の出力を返す場合があります。コマンド全体が完了する前に NX-API クライアントが出力の処理を開始するために、NX-API は show コマンドの出力チャンクをサポートしています。 次の設定を有効または無効にできます。 0 : チャンク出力しません。 1 : チャンク出力します。 (注) - チャンクをサポートするのは show コマンドだけです。一連の show コマンドが入力されると、最初のコマンドだけがチャンクされて返されます。 出力メッセージの形式は XML で、これがデフォルトです。＜または＞などの特殊文字は、有効な XML メッセージを形成できるよう変換されます（＜は<に変換され、<>は>に変換されます）。 XML SAX を使用して、チャンクされた出力を解析できます。 - チャンクが有効な場合、メッセージ形式は XML に制限されます。チャンクが有効な場合、JSON 出力形式はサポートされません。 (注) チャンクが有効になっている場合、現在サポートされている最大メッセージサイズは、チャンク出力で 200 MB です。
<i>roll_back</i>	コンフィギュレーションロールバック オプションを指定します。次のいずれかのオプションを指定します。 - Stop-on-error : 最初に失敗した CLI で停止します。 - Continue-on-error : 他の CLI を無視して続行します。 - Rollback-on-error : システム設定を以前の状態にロールバックします。 (注) rollback-on-error オプションは、Cisco MDS NX-OS リリース 9.2(2) から削除されました。

NX-API リクエスト要素	説明
検証	構成検証設定この要素を使用すると、スイッチに適用する前にコマンドを検証できます。これにより、設定を適用する前に、設定の整合性（必要なハードウェアリソースの可用性など）を確認できます。[検証タイプ（Validation Type）] ドロップダウン リストから検証タイプを選択します。 • Validate-Only：設定を検証しますが、設定は適用しません。 • Validate-and-Set：設定を検証し、検証が成功した場合はスイッチに構成を適用します。
ロック	構成の排他ロックを指定できます。これにより、このロックが保持されている場合、他の管理エージェントまたはプログラミングエージェントは構成を変更できません。
<i>sid</i>	セッションID要素は、応答メッセージがチャンクされている場合にのみ有効です。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 <i>sid</i> と一致する <i>sid</i> を指定する必要があります。
<i>input</i>	入力 は 1 つのコマンドまたは複数のコマンドです。ただし、異なるメッセージタイプに属するコマンドを混在させてはなりません。たとえば、show コマンドは cli_show メッセージフォーマットであり、cli_conf メッセージフォーマットではサポートされません。 （注） 複数のコマンドは、セミコロン (;) で区切ることができます。（; は、単一のブランク文字で囲む必要があります。） 以下は、複数のコマンドの例です。 • cli_show show version ; show interface brief ; show vsan • cli_conf interface fc4/1 ; no shut

NX-API リクエスト要素	説明
<code>output_format</code>	使用可能な出力メッセージフォーマットは次のとおりです： • <code>xml</code>：XML フォーマットでの出力を指定します。 • <code>json</code>：JSON フォーマットでの出力を指定します。 • <code>json-rpc</code>：JSON-RPC フォーマットでの出力を指定します。 (注) Cisco MDS 9000 デバイスの CLI は XML 出力をサポートしています。つまり、JSON 出力は XML から変換されます。変換はスイッチで処理されます。 計算のオーバーヘッドを管理するために、JSON 出力を行うかどうかは、出力サイズによって決定されます。出力が 1MB を超える場合、出力は XML 形式で返されます。出力がチャックされている場合、XML 出力のみがサポートされます。 HTTP または HTTPS ヘッダーの <code>content-type</code> ヘッダーは、応答フォーマット (XML、JSON、または JSON-RPC) のタイプを示します。

NX-API 応答要素

次の表は、CLI コマンドに応答する NX-API 要素を示しています：

表 2: NX-API 応答要素

NX-API 応答要素	説明
<code>version</code>	NX-API バージョン。
<code>type</code>	実行するコマンドのタイプ。
<code>sid</code>	応答のセッション識別子。この要素は、応答メッセージがチャックされている場合にのみ有効です。
<code>outputs</code>	すべてのコマンド出力を囲むタグ。 複数のコマンドが <code>cli_show</code> または <code>cli_show_ascii</code> コマンドタイプのいずれかである場合、各コマンド出力は単一の出力タグで囲まれます。 コマンドタイプが <code>cli_conf</code> の場合、<code>cli_conf</code> コマンドにはコンテキストが必要なため、すべてのコマンドで単一のタグが出力されます。

NX-API 応答要素	説明
出力	単一のコマンド出力の出力を囲むタグ。 cli_conf コマンドタイプの場合、この要素にはすべてのコマンドの出力が含まれます。
input	リクエストで指定された 1 つのコマンドを囲むタグ。この要素は、要求入力要素を適切な応答出力要素に関連付けるのに役立ちます。
本文	コマンド応答の本文。
コード	コマンドの実行から返されたエラー コード。 NX-API は、HTTP ステータス コード レジストリ (http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml) で説明されている標準規格の HTTP 原因コードを使用します。
msg	返された原因コードに関連付けられたエラー メッセージ。

NX-API 応答コードの表

次に、NX-API 応答に関連する NX-API エラー、エラー コード、およびメッセージを示します。

表 3: NX-API 応答コード

[NX-API 応答 (NX-API Response)]	コード	メッセージ
成功	200	成功。
CUST_OUTPUT_PIPED	204	要求により、出力は別の場所にパイプされます。
CHUNK_ALLOW_ONE_CMD_ERR	400	チャンクは1つのコマンドにのみ許可されます。
CLI_CLIENT_ERR	400	CLI の実行エラー
CLI_CMD_ERR	400	CLI コマンド エラーの入力。
IN_MSG_ERR	400	要求メッセージが無効です。
NO_INPUT_CMD_ERR	400	入力コマンドがありません。
PERM_DENY_ERR	401	権限が拒否されました。
CONF_NOT_ALLOW_SHOW_ERR	405	構成モードでは、 show コマンドを使用できません。

SHOW_NOT_ALLOW_CONF_ERR	405	表示モードでは構成できません。
EXCEED_MAX_SHOW_ERR	413	連続する show コマンドの最大数を超過しました。最大値は 10 です。
MSG_SIZE_LARGE_ERR	413	応答サイズが大きすぎます。
BACKEND_ERR	500	バックエンド処理エラー。
FILE_OPER_ERR	500	システム内部ファイル操作エラー。
LIBXML_NS_ERR	500	システムの内部 LIBXML NS エラー。
LIBXML_PARSE_ERR	500	システムの内部 LIBXML 解析エラー。
LIBXML_PATH_CTX_ERR	500	システムの内部 LIBXML パス コンテキストエラー。
MEM_ALLOC_ERR	500	システムの内部メモリ割り当てエラー。
USER_NOT_FOUND_ERR	500	入力またはキャッシュからユーザーが見つかりません。
XML_TO_JSON_CONVERT_ERR	500	XML から JSON への変換エラー。
CHUNK_ALLOW_XML_ONLY_ERR	501	チャンクはXML出力のみを許可します。
JSON_NOT_SUPPORTED_ERR	501	大量の出力のため、JSONはサポートされていません。
MSG_TYPE_UNSUPPORTED_ERR	501	メッセージタイプはサポートされていません
PIPE_OUTPUT_NOT_SUPPORTED_ERR	501	パイプ操作はサポートされていません。
PIPE_XML_NOT_ALLOWED_IN_INPUT	501	入力でパイプ XML は許可されていません。
RESP_BIG_JSON_NOT_ALLOWED_ERR	501	応答の出力量が多い。JSONはサポートされません。
STRUCT_NOT_SUPPORTED_ERR	501	構造化出力はサポートされていません。
ERR_UNDEFINED	600	未定義。

デフォルト設定

次の表に、Cisco MDS リリース バージョンの HTTP および HTTPS のデフォルト設定を示します。

Cisco MDS NX-OS リリース	HTTP	HTTPS
Cisco MDS NX-OS リリース 8.2(2) 以前	有効	無効
Cisco MDS NX-OS リリース 8.3(1) Cisco MDS NX-OS リリース 8.3(2)	有効	有効
Cisco MDS NX-OS リリース 8.4(1) 以降のリリース	無効	有効

表 4: サポートされる HTTP および HTTPS ポート (34 ページ) の表に、Cisco MDS リリースバージョンでサポートされている HTTP および HTTPS ポートを示します。

表 4: サポートされる HTTP および HTTPS ポート

Cisco MDS NX-OS リリース	HTTP ポート (HTTP Port)	HTTPS ポート (HTTPS Port)
Cisco MDS NX-OS リリース 8.2(1) 以前	8080	443
Cisco MDS NX-OS リリース 8.3(1) 以降	8080	8443

その他の参考資料

このセクションでは、NX-API の実装に関する追加情報について説明します。

- [NX-API DevNet コミュニティ](#)
- [MDS NX-API リファレンス ガイド](#)
- [NX-API Github \(NX-OS プログラマビリティ スクリプト\)](#)
- [CISCO DCNM API リファレンス ガイド](#)

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。