Cisco MDS 9000 シリーズ プログラマビリティ ガイド、リリース 9.x

最終更新：2025 年 5 月 16 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次

はじめに

ここでは、『Cisco MDS 9000 Series Configuration Guideを使用している対象読者、構成、および表記法について説明します。また、関連マニュアルの入手方法についても説明します。次のセクションを含んでいます：

対象読者

このマニュアルは、Cisco MDS 9000 シリーズ スイッチの設置、構成、および維持に携わるネットワーク管理者を対象としています。

表記法

コマンドの説明では、次の表記法を使用しています。

表記法	説明
bold	太字の文字は、表示どおりにユーザが入力するコマンドおよびキーワードです。
italic	イタリック体の文字は、ユーザが値を入力する引数です。
[x]	省略可能な要素（キーワードまたは引数）は、角かっこで囲んで示しています。
[x | y]	いずれか1つを選択できる省略可能なキーワードや引数は、角カッコで囲み、縦棒で区切って示しています。

表記法	説明
{x | y}	必ずいずれか1つを選択しなければならない必須キーワードや引数は、波かっこで囲み、縦棒で区切って示しています。
[x {y | z}]	角かっこまたは波かっこが入れ子になっている箇所は、任意または必須の要素内の任意または必須の選択肢であることを表します。角かっこ内の波かっこと縦棒は、省略可能な要素内で選択すべき必須の要素を示しています。
`variable`	ユーザが値を入力する変数であることを表します。イタリック体が使用できない場合に使用されます。
string	引用符を付けない一組の文字。string の前後には引用符を使用しません。引用符を使用すると、その引用符も含めて string とみなされます。

例では、次の表記法を使用しています。

表記法	説明
`screen フォント`	スイッチが表示する端末セッションおよび情報は、スクリーンフォントで示しています。
太字の `screen` フォント	ユーザが入力しなければならない情報は、太字のスクリーンフォントで示しています。
イタリック体の *screen* フォント	ユーザが値を指定する引数は、イタリック体の screen フォントで示しています。
< >	パスワードのように出力されない文字は、山カッコ（< >）で囲んで示しています。
[ ]	システム プロンプトに対するデフォルトの応答は、角カッコで囲んで示しています。
!、#	コードの先頭に感嘆符（!）またはポンド記号（#）がある場合には、コメント行であることを示します。

このマニュアルでは、次の表記法を使用しています。

（注） 「注釈」です。役立つ情報やこのマニュアルに記載されていない参照資料を紹介しています。

注意 「要注意」の意味です。機器の損傷またはデータ損失を予防するための注意事項が記述されています。

警告 安全上の重要事項

「危険」の意味です。人身事故を予防するための注意事項が記述されています。装置の取り扱い作業を行うときは、電気回路の危険性に注意し、一般的な事故防止策に留意してください。各警告の最後に記載されているステートメント番号を基に、装置に付属の安全についての警告を参照してください。

これらの注意事項を保管しておいてください。

マニュアルに関するフィードバック

このマニュアルに関する技術的なフィードバック、または誤りや記載もれなどお気づきの点がございましたら、mds-docfeedback@cisco.comよりご連絡ください。ご協力をよろしくお願いいたします。

関連資料

Cisco MDS 9000 シリーズ スイッチ全体のマニュアル セットは、次の URL にあります:

https://www.cisco.com/c/en/us/support/storage-networking/mds-9000-nx-os-san-os-software/series.html

ドキュメント ロードマップ

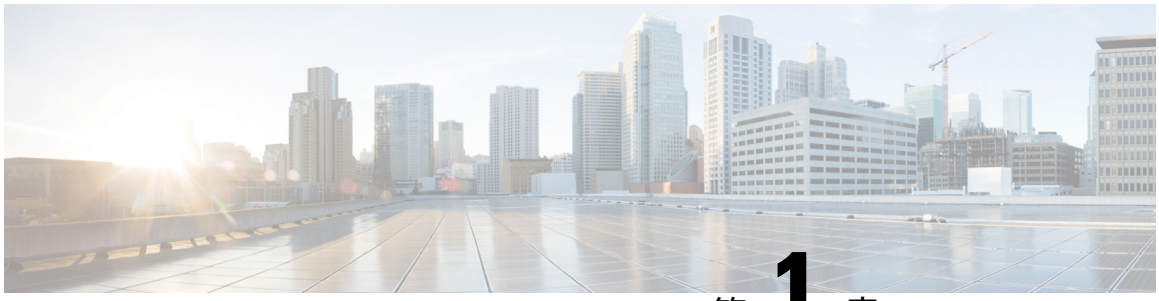
https://www.cisco.com/c/en/us/td/docs/storage/san_switches/mds9000/roadmaps/rel90.html

通信、サービス、およびその他の情報

- シスコからタイムリーな関連情報を受け取るには、Cisco Profile Manager でサインアップしてください。
- 重要な技術によって求めるビジネス成果を得るには、Cisco Services [英語] にアクセスしてください。
- サービス リクエストを送信するには、Cisco Support にアクセスしてください。
- 安全で検証済みのエンタープライズクラスのアプリケーション、製品、ソリューション、およびサービスを探して参照するには、Cisco DevNet にアクセスしてください。
- 一般的なネットワーキング、トレーニング、認定関連の出版物を入手するには、Cisco Press にアクセスしてください。
- 特定の製品または製品ファミリの保証情報を探すには、Cisco Warranty Finder にアクセスしてください。

Cisco バグ検索ツール

Cisco Bug Search Tool（BST）は、シスコ製品とソフトウェアの障害と脆弱性の包括的なリストを管理する Cisco バグ追跡システムへのゲートウェイとして機能する、Web ベースのツールです。BST は、製品とソフトウェアに関する詳細な障害情報を提供します。



第 1 章

新機能および変更された機能に関する情報

新機能および変更された機能に関する情報

次の表は、*Cisco MDS Nexus 9000* シリーズ プログラマビリティ ガイドに記載されている新機能および変更された機能を要約したものです。それぞれの説明が記載されているセクションも併記されています。

表 *1*: 新機能および変更された機能

新機能/拡張された機能	説明	導入された Cisco MDS NX-OS のリリース	トピックが記載されている箇所
Python	Python バージョン 2.7.8 のサポートが削除されました。	9.2（2）	Python API （39 ページ）
NX-API	暗号化された秘密キーを使用した NX-API 証明書のインストールとトラストポイント証明書の使用がサポートされました。	8.5(1)	NX-API （5 ページ）
Python	Python 3.0 のサポートが追加されました。	8.4（2）	Python API （39 ページ）
Ansible	Ansible のサポートが追加されました。	8.4(1)	Ansible （51 ページ）

新機能および変更された機能に関する情報

新機能/拡張された機能	説明	導入された Cisco MDS NX-OS のリリース	トピックが記載されている箇所
NX-API	**cli_show_array** コマンドタイプのサポートが追加されました。 NX-API 開発者サンドボックスが変更されました。**コマンドリファレンス**のオプションが追加されました。 Java および JavaScript コード形式のサポートが追加されました。	8.4(1)	NX-API （5 ページ）
NX-API	NX-API over HTTPS 自己署名証明書の有効期限は、NX-OS 8.3(1) リリースで変更されました。	8.3(1)	NX-API （5 ページ）

新機能および変更された機能に関する情報

次の表は、*Cisco MDS Nexus 9000* シリーズ プログラマビリティ ガイドに記載されている新機能および変更された機能を要約したものです。それぞれの説明が記載されているセクションも併記されています。

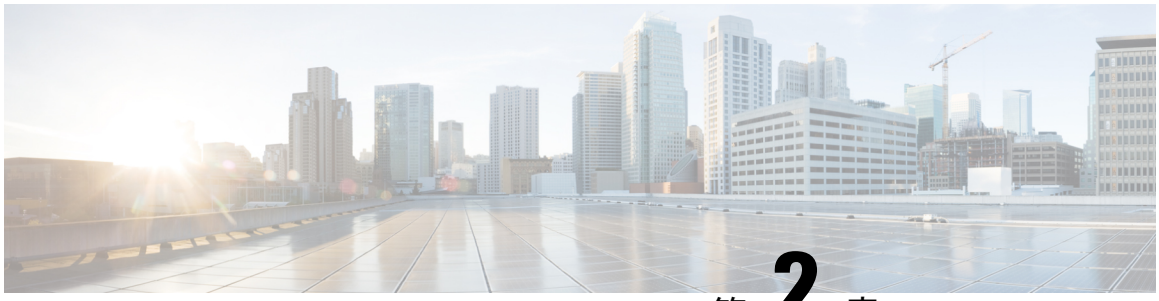
表 *2:* 新機能および変更された機能

新機能/拡張された機能	説明	導入された Cisco MDS NX-OS のリリース	トピックが記載されている箇所
Python	Python バージョン 2.7.8 のサポートが削除されました。	9.2（2）	Python API （39 ページ）
NX-API	暗号化された秘密キーを使用した NX-API 証明書のインストールとトラストポイント証明書の使用がサポートされました。	8.5(1)	NX-API （5 ページ）

新機能および変更された機能に関する情報

新機能/拡張された機能	説明	導入された **Cisco MDS NX-OS** のリリース	トピックが記載されている箇所
Python	Python 3.0 のサポートが追加されました。	8.4（2）	Python API （39 ページ）
Ansible	Ansible のサポートが追加されました。	8.4(1)	Ansible （51 ページ）
NX-API	**cli_show_array** コマンドタイプのサポートが追加されました。 NX-API 開発者サンドボックスが変更されました。**コマンドリファレンス**のオプションが追加されました。 Java および JavaScript コード形式のサポートが追加されました。	8.4(1)	NX-API （5 ページ）
NX-API	NX-API over HTTPS 自己署名証明書の有効期限は、NX-OS 8.3(1) リリースで変更されました。	8.3(1)	NX-API （5 ページ）

新機能および変更された機能に関する情報



第 2 章

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

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 ネットワーキングの自動化が容易になります。

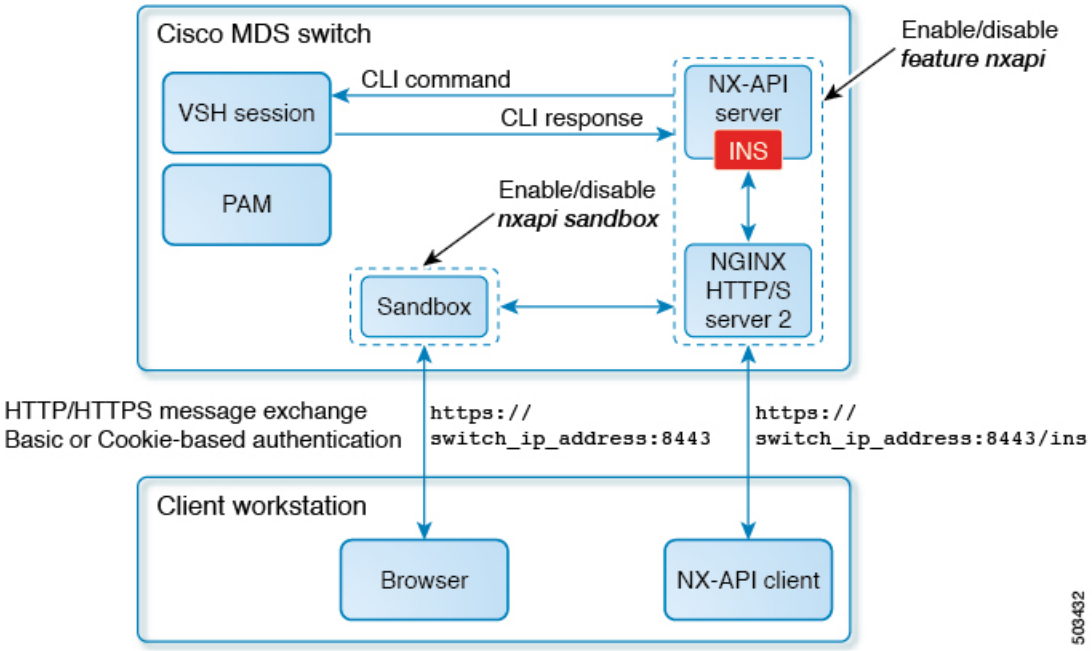
Cisco MDS NX-API は、特定の **show** コマンドと非インタラクティブな構成コマンドをサポートしています。

（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



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ーが、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 （18 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 （36 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 （37 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されるパスワードです。
- *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** コマンドで使用される **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るため、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/idcert.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_tmstmp":  "09/08/2020 09:42:15",
                "isan_file_name":       "bootflash:///sky-sep14-02",
                "isan_cmpl_time":       " 12/20/2020 12:00:00",
                "isan_tmstmp":  "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_uptm_days":       0,
                "kern_uptm_hrs":        1,
                "kern_uptm_mins":       25,
                "kern_uptm_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_tmstmp>01/08/2020 18:27:03</kick_tmstmp>
     <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_tmstmp>01/14/2020 05:36:15</isan_tmstmp>
     <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_tmstmp": "01/08/2020 18:27:03",
  "isan_file_name": "bootflash:///m9700-sf3ek9-mzg.8.4.2.191.bin",
  "isan_cmpl_time": "2/5/2020 12:00:00",
  "isan_tmstmp": "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_uptm_days": 19,
  "kern_uptm_hrs": 23,
  "kern_uptm_mins": 16,
  "kern_uptm_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 supp
ort: http://www.cisco.com/tac\nDocuments: http://www.cisco.com/en/US/products/ps
9372/tsd_products_support_series_home.html\nCopyright (c) 2002-2020, Cisco Syste
ms, Inc. All rights reserved.\nThe copyrights to certain works contained herein
are owned by\nother third parties and are used and distributed under license.\nS
ome 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.bi
n",
"kick_cmpl_time": " 7/11/2020 12:00:00",
"kick_tmstmp": "06/20/2020 20:50:09",
"isan_file_name": "bootflash:///m9100-s5ek9-mz.8.4.2a.bin",
"isan_cmpl_time": " 7/11/2020 12:00:00",
"isan_tmstmp": "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_uptm_days": 7,
"kern_uptm_hrs": 1,
"kern_uptm_mins": 13,
"kern_uptm_secs": 0,
"rr_usecs": 362070,
"rr_ctime": "Mon Sep 28 07:43:36 2020",
```

```
"rr_reason": "Reset due to upgrade",
"rr_sys_ver": "8.4(2b)",
"rr_service": "",
"manufacturer": "Cisco Systems, Inc."
}
```

NX-API 開発者サンドボックス

NX-API 開発者サンドボックスは、NX-API 要求を作成し、応答を受信するために使用される、Cisco が開発した Web ベースのユーザー インターフェイスです。要求は、**show** コマンド、および非インタラクティブ コンフィギュレーション コマンドの形式です。

NX-API サンドボックスの構成

1. NX-API 機能を有効にし、構成します。NX-API 機能を有効にして構成する方法については、NX-API CLI の構成 （10 ページ）セクションを参照してください。
2. NX-API サンドボックスを有効にします：

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

NX-API 設定を表示するには、**show nxapi** コマンドを使用します。

```
switch# show nxapi

NX-API:       Enabled         Sandbox:      Enabled
HTTP Port:    Disabled        HTTPS Port:  8443

Certificate Information:
    Issuer:   C=US, ST=CA, L=San Jose, O=Cisco Systems Inc., OU=dcnxos, CN=nxos
    Expires:  Nov 26 09:26:12 2019 GMT
    Content:  -----BEGIN CERTIFICATE-----
MIIDpzCCAo+gAwIBAgIJAObmdczeHJL8MA0GCSqGSIb3DQEBCwUAMGoxCzAJBgNV
BAYTAlVTMQswCQYDVQQIDAJDQTERMA8GA1UEBwwIU2FuIEpvc2UxGzAZBgNVBAoM
EkNpc2NvIFN5c3RlbXMgSW5jLjEPMA0GA1UECwwGZGNueG9zMQ0wCwYDVQQDDARu
eG9zMB4XDTE5MTEyNTA5MjYxMloXDTE5MTEyNjA5MjYxMlowajELMAkGA1UEBhMC
VVMxCzAJBgNVBAgMAkNBMREwDwYDVQQHDAhTYW4gSm9zZTEbMBkGA1UECgwSQ2lz
Y28gU3lzdGVtcyBJbmMuMQ8wDQYDVQQLDAZkY254b3MxDTALBgNVBAMMBG54b3Mw
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDreVT3LAYarHxZxELwNwst
ZQqlmah5PJHBGUx+3YQcRi8v8wrEsRI7bZrQgDzXkFEH9yroJUghdc0nkM1GYHNF
avbj4qRaEveRejtGZNMPi11tAOWbRsU0ldxODV3+SeG/A220Bal58JzJjje5wyi8
Wu8UR8w4Lb32GYXI8ifBtlU0LrNsS0pE68yJt782y8IJIAEjGWX3L8dle4qwRqXg
6GBDPbwKFCvi+NX8JN48olONASSHXGdcmZSfiYUNKPD7+AwjG/luxSyIqnFW2w06
zwQLoIbXJx7zvl0Adt3H4ZnaRZOG5UvsWLgEdJZJkaqQPe0+7cpwcubN9/PJlO0l
AgMBAAGjUDBOMB0GA1UdDgQWBBTUI4IqOqmoKy5LEjygiJzd2nEGWzAfBgNVHSME
GDAWgBTUI4IqOqmoKy5LEjygiJzd2nEGWzAMBgNVHRMEBTADAQH/MA0GCSqGSIb3
DQEBCwUAA4IBAQC+7TPfDAz1t4yvG4rpypinU2Plp2FOBRdU85CIVREIplbRX8Vv
VMXYySnrKDjVRPcWtY+EtDW91BfS2I2usHkiKcnOYazDoxpamFI3D6lmb82JAAqG
NMV56hIJAYMpVqfbI+vtC93NR3F2fLD8/Hm/X4L8U6kdu2o+vgqYtc4OOl87lhJS
R8xA2N9kihOA1FUMVL89cFgRvxONjAyQImTB8uWl1stUGpH2kE3dABHC1sbZ1dNw
2/OxpfGnj1Qjvi4wKqjGpX/Kqc0AIx2zsKEX9hpPMQK/wlaRg8hlNCXJzzTQ7IVX
7PWJSqn7gpfyZIgZ9JQQ/WieCH32mQ2xGMDD
-----END CERTIFICATE-----
```

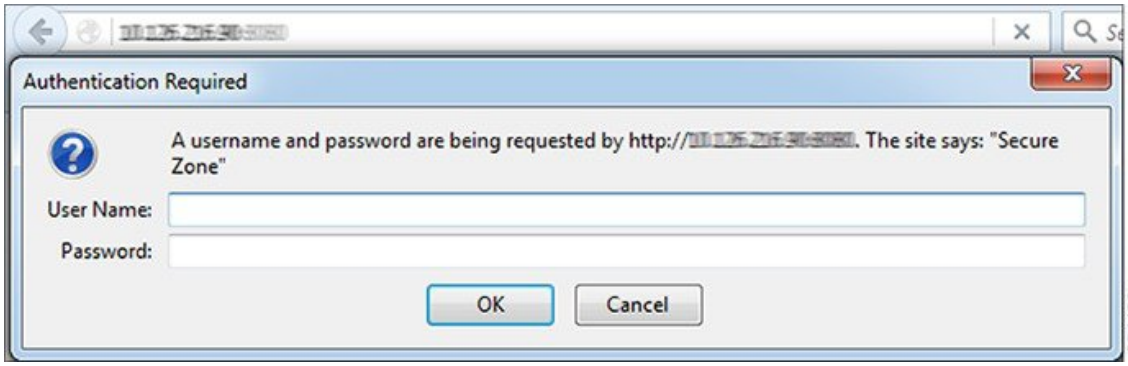
NX-API 開発者サンドボックスにアクセスするには、次の手順を実行します：

（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



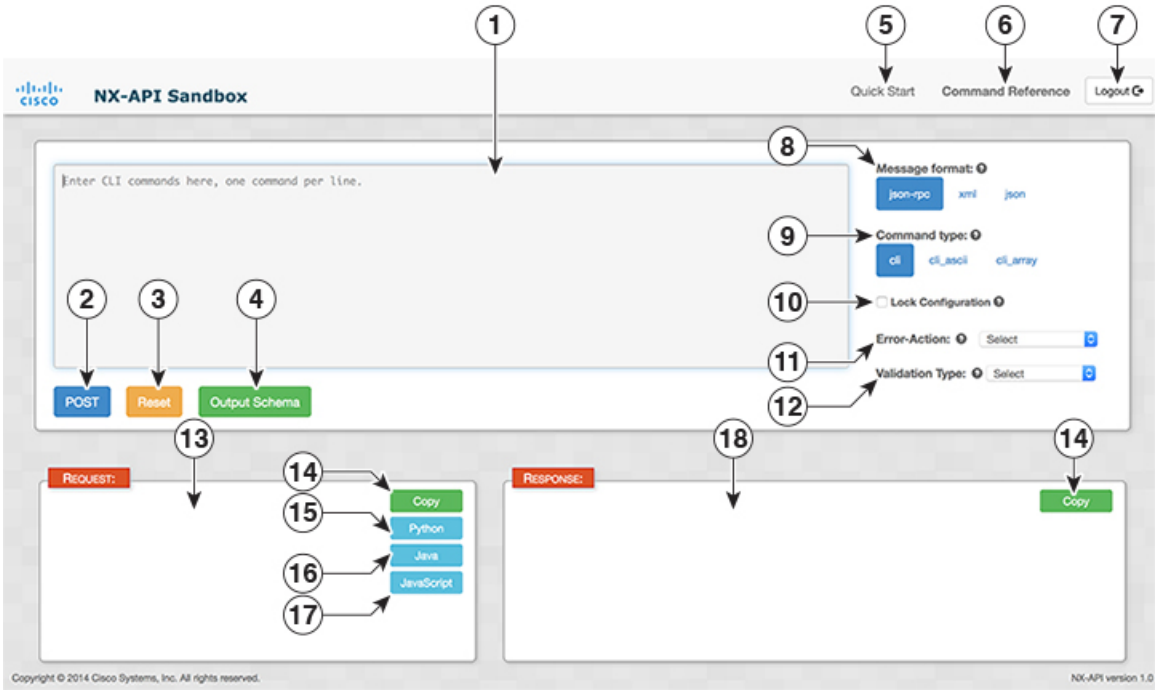
2. 適切なクレデンシャルでログインします。

   [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の [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。

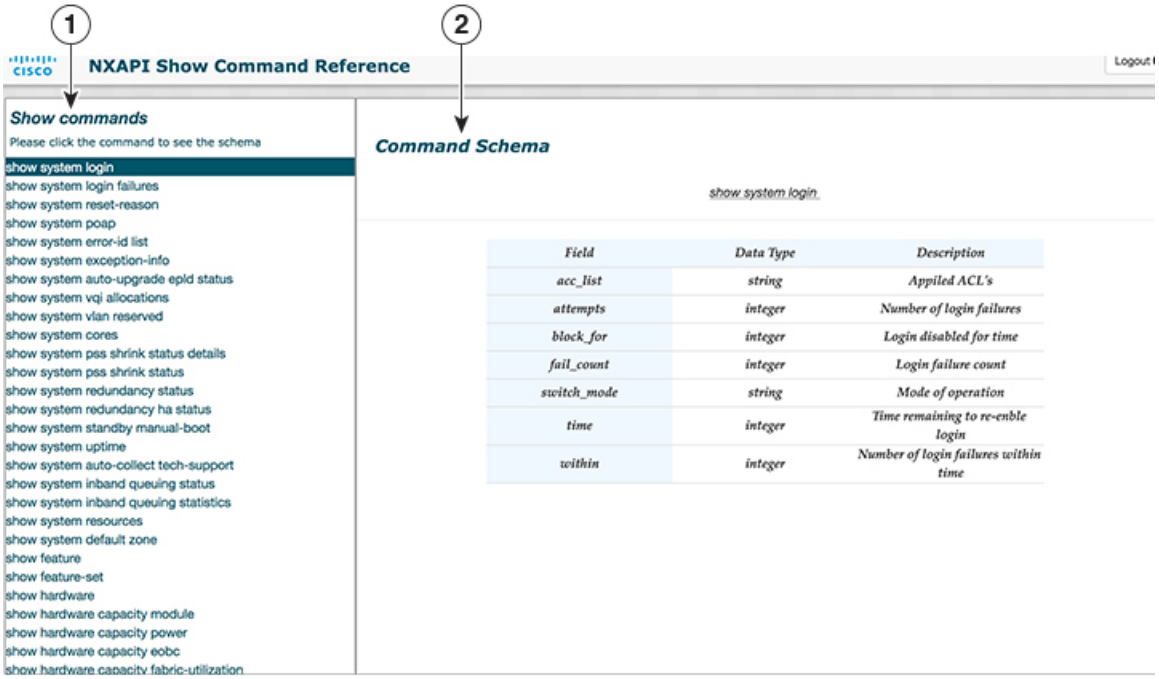
コマンドとそれに対応する出力をクリアし、ページをリセットするには、**[リセット（Reset）]**をクリックします。

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 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 pwwn 10:00:00:23:45:67:89:ab
member pwwn 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 pwwn 10:00:00:23:45:67:89:ab ;member pwwn
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 pwwn 10:00:00:23:45:67:89:ab</input>
      <code>200</code>
      <msg>Success</msg>
    </output>
    <output>
      <body/>
      <input>member pwwn 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>
```

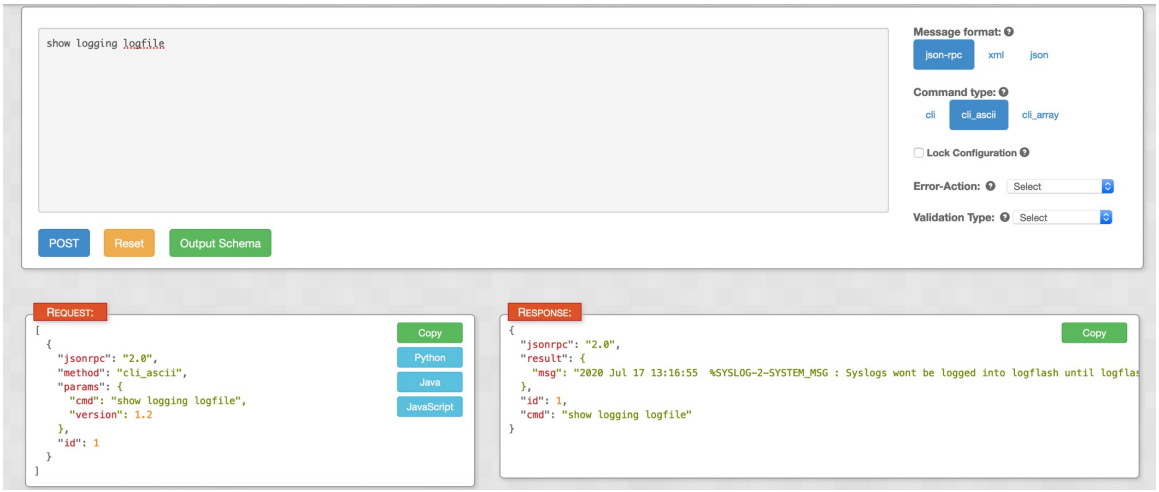
応答：

```
<?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。



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 *3 : 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
type	

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。
roll_back	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。
sid	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 *sid* と一致する *sid* を指定する必要があります。
input	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
output_format	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： • xml：XML フォーマットでの出力を指定します。 • json：JSON フォーマットでの出力を指定します。 • json-rpc：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 1 MB を超える場合、出力は XML 形式で返されます。出力がチャンクされている場合、XML 出力のみがサポートされます。 HTTP または HTTPS ヘッダーの content-type 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 *4 : NX-API* 応答要素

NX-API 応答要素	説明
version	NX-API バージョン。
type	実行するコマンドのタイプ。
sid	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ンクされている場合にのみ有効です。
outputs	すべてのコマンド出力を囲むタグ。 複数のコマンドが cli_show または cli_show_ascii 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 cli_conf の場合、cli_conf 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 *5 : 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えました。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 6 : サポートされる HTTP および HTTPS ポート （38 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。

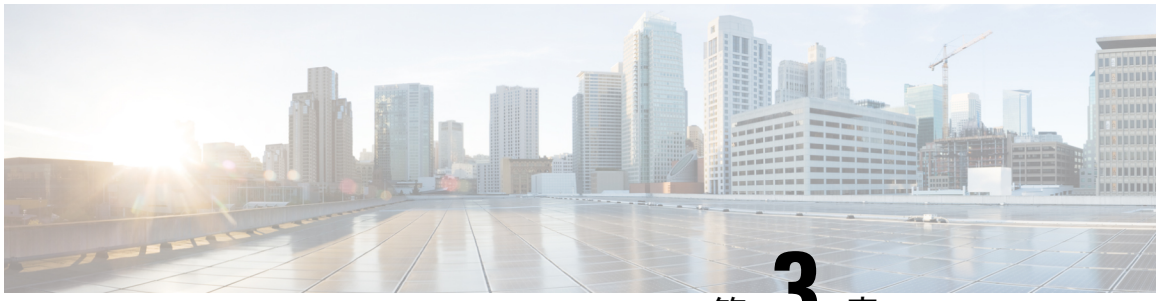
表 *6 :* サポートされる *HTTP* および *HTTPS* ポート

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



第 3 章

Python API

[Python API について（About the Python API）]

Python は簡単に習得できる、強力なプログラミング言語です。効率的で高水準なデータ構造を持ち、オブジェクト指向プログラミングに対してシンプルで効果的なアプローチを取っています。Python の構文と動的な型指定は、インタープリタ型という特長と相まって、スクリプティングと高速アプリケーション開発のための理想的な言語にしています。Python の Web サイト、www.python.org には、サードパーティが無償で提供している多数の Python モジュール、プログラム、ツールのディストリビューションとそれらへのリンク、さらに追加のドキュメンテーションが掲載されています。

Python インタープリタは、標準および Cisco Python モジュールとともに Cisco MDS NX-OS コマンドラインインターフェイス（CLI）で使用できます。インタラクティブモードと非インタラクティブ（スクリプト）モードの両方がサポートされています。これにより、繰り返しタスクを実行するよう MDS デバイスをプログラムで制御できます。これを活用できる例としては、PowerOn Auto Provisioning（POAP）スクリプトと Embedded Event Manager（EEM）アクションがあります。また、SAN 分析機能のオーバーレイ CLI で分析データをフォーマットするためにも使用されます。

サポートされるバージョン

表 7 : MDS プラットフォームの Python バージョン履歴 （40 ページ） に、Cisco MDS スイッチでの Python のマイルストーンを示します。各プラットフォームでサポートされているバージョンとデフォルトのバージョンがいつ変更されたかが表示されます。

表 *7: MDS* プラットフォームの *Python* バージョン履歴

リリース	Cisco NX-OS MDS プラットフォーム							
	9132T	**9148S**	**9148T**	**9220i**	**9250i**	**9396S**	**9396T**	**9700**
6.2(29)	—	—	—	—	—	—	—	2.7
8.3(1)	2.7.8	—	2.7.8	—	—	—	2.7.8	2.7.8
8.4（2）	2.7.8* 3.7.3	—	2.7.8* 3.7.3	—	—	—	2.7.8* 3.7.3	2.7.8* 3.7.3
8.5(1)	2.7.8* 3.7.3	—	2.7.8* 3.7.3	2.7.8* 3.7.3	—	—	2.7.8* 3.7.3	2.7.8* 3.7.3
9.2(1)	2.7.8* 3.7.3	—	2.7.8* 3.7.3	2.7.8* 3.7.3	—	—	2.7.8* 3.7.3	2.7.8* 3.7.3
9.2（2）	3.7.3*	—	3.7.3*	3.7.3*	—	—	3.7.3*	3.7.3*

* はデフォルトの Python バージョンを示しています。

（注） Python API は、Cisco MDS 9148S、Cisco MDS 9250i、Cisco MDS 9396S などの Cisco MDS 16 Gbps ファブリック スイッチではサポートされていません。

Python の使用

ここでは、Python スクリプトの作成と実行の方法について説明します。

Cisco Python パッケージ

Cisco MDS NX-OS は、インターフェイス、VLAN、ルートなど、多くのコア ネットワーク デバイス モジュールへのアクセスを可能にする Cisco Python パッケージを提供します。**help()** コマンドを入力すると、Cisco Python パッケージの詳細を表示できます。モジュール内のクラスとメソッドに関する追加情報を取得するには、特定のモジュールに対して help コマンドを実行します。たとえば、**help**（*cisco.interface*）は、cisco.interface モジュールのプロパティを表示します。

次の例は、Cisco python パッケージに関する情報を表示する方法を示します。

```
>>> import cisco
>>> help(cisco)
Help on package cisco:

NAME
    cisco
```

```
DESCRIPTION
    ###############################################################################
    #
    #      File:   cli.py
    #      Name:
    #
    #      Description:
    #
    # Copyright (c) 2015-2017, 2019-2020 by cisco Systems, Inc.
    # All rights reserved.
    #
    ###############################################################################

PACKAGE CONTENTS
    acl
    bgp
    buffer_depth_monitor
    check_port_discards
    cisco_secret
    feature
    history
    interface
    ipaddress
    key
    line_parser
    mac_address_table
    nxcli
    ospf
    routemap
    routes
    section_parser
    ssh
    system
    tacacs
    transfer
    vlan
    vrf

CLASSES
    builtins.object
```

CLI コマンド API の使用

Python プログラミング言語は、CLI コマンドを実行できる 3 つの API を使用します。API は Python CLI モジュールから利用できます。

これらの API については、次の表で説明します。**from cli import *** コマンドを使用して API を有効にする必要があります。これらの API の引数は CLI コマンドの文字列です。Python インタープリタ経由で CLI コマンドを実行するには、次の API のいずれかの引数文字列として CLI コマンドを入力します。

表 *8 : CLI* コマンド *API*

API	説明
cli() 例： `string = cli ("cli-command")`	制御文字/特殊文字を含む CLI コマンドの未処理の出力を返します。 （注） インタラクティブな Python インタープリタは、制御文字/特殊文字を「エスケープ」して出力します。改行は「\n」として出力されるため、結果が読みにくくなる場合があります。**clip()** API は、判読性が高い結果を出力します。
clid() 例： `json_string = clid ("cli-command")`	XML をサポートする CLI コマンドの場合、この API は JSON 出力を返します。 （注） XML が使用されていない場合は、例外がスローされます。 この API は、show コマンドの出力の検索時に使用すると便利な場合があります。

clip()
例：
`clip ("cli-command")`
	CLI コマンドの出力を直接 stdout に出力し、Python には何も返されません。
	（注）
	`clip ("cli-command")`
	と同等です（is equivalent to）
	`r=cli("cli-command")`
	`print r`

2 つ以上のコマンドを個別に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続しません。

次の例では、最初のコマンドの状態が 2 番目のコマンドで持続しないため、2 番目のコマンドが失敗します。

```
>>> cli("conf t")
>>> cli("interface fc4/1")
```

2 つ以上のコマンドを同時に実行すると、その状態は 1 つのコマンドから後続のコマンドまで持続します。

次の例では、2 番目と 3 番目のコマンドの状態が持続するため、2 番目のコマンドは成功しています。

```
>>> cli("conf t ; interface fc4/1 ; shut")
```

（注） 例に示すように、コマンドは「;」で区切られます。（; は、単一のブランク文字で囲む必要があります。）

CLI からの Python インタープリタの呼び出し

次に、CLI から Python を呼び出す方法の例を示します：

（注）
- Python インタープリタのプロンプトは「>>>」または「…」で表示されます。
- Cisco MDS NX-OS リリース 7.3(x) 以降のリリースでは、Cisco MDS 9700 シリーズ スイッチの特権 EXEC モードでのみ Python インタープリタを呼び出すことができます。
- Cisco MDS NX-OS リリース 9.2(2) 以降では、python コマンドは Python 3.0 を実行します。

次に、CLI から Python2 .7.5 を呼び出す方法の例を示します：

```
switch# python

Python 2.7.5 (default, Jun  3 2016, 03:57:06)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from cli import *
>>> cli("show clock")
'Time source is NTP\n06:32:20.023 UTC Tue Jan 31 2017\n'
>>> exit()
```

次に、CLI から Python3 を呼び出す方法の例を示します：

```
switch# python3

Python 3.7.3 (default, Aug 26 2019, 23:20:10)
[GCC 4.6.3] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cli import *
>>> cli("show clock")
'Time source is NTP\n07:46:50.923 UTC Tue Apr 28 2020\n'
```

表示フォーマット

次に、Python API を使用したさまざまな表示フォーマットを示します：

例 1：

```
>>> from cli import *
>>> cli("conf ; interface fc1/1")
''
clip('where detail')
  mode:
  username:            admin
```

```
>>>
```

例 2：

```
>>> from cli import *
>>> cli("conf ; interface fc1/1")
''
>>> cli('where detail')
'  mode:                \n  username:            admin\n'
>>>
```

例 3：

```
>>> from cli import *
>>> cli("conf ; interface fc1/1")
''
>>> r = cli('where detail') ; print(r)
  mode:
  username:            admin

>>>
```

例 4：

```
>>> from cli import *
>>> import json
>>> out=json.loads(clid('show version'))
>>> for k in out.keys():
...    print("%30s = %s" % (k, out[k]))
...
                    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) 2002-2022, 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
                   bios_ver_str = 2.12.0
              kickstart_ver_str = 9.2(2)
                    sys_ver_str = 9.2(2)
                 bios_cmpl_time = 05/26/2021
                 kick_file_name = bootflash:///m9700-sf4ek9-kickstart-mz.9.2.2.bin
                 kick_cmpl_time = 1/1/2022 12:00:00
                    kick_tmstmp = 01/20/2022 22:42:00
                 isan_file_name = bootflash:///m9700-sf4ek9-mz.9.2.2.bin
                 isan_cmpl_time = 1/1/2022 12:00:00
                    isan_tmstmp = 01/21/2022 00:12:45
                     chassis_id = MDS 9706 (6 Slot) Chassis
                      module_id = Supervisor Module-4
                       cpu_name = Intel(R) Xeon(R) CPU D-1548 @ 2.00GHz
                         memory = 14146356
                       mem_type = kB
                   proc_board_id = JAE22320AXJ
                       host_name = sw184-9706
                 bootflash_size = 3932160
                     slot0_size = 0
                kern_uptm_days = 5
```

```
                 kern_uptm_hrs = 21
                kern_uptm_mins = 39
                kern_uptm_secs = 27
                      rr_usecs = 699796
                      rr_ctime = Tue Jan 25 10:40:02 2022
                     rr_reason = Reset Requested by CLI command reload
                    rr_sys_ver = 9.2(2)
                    rr_service = None
                  manufacturer = Cisco Systems, Inc.
>>>
```

非インタラクティブ Python

Python スクリプト名を引数として Python CLI コマンドで使用することで、Python スクリプトを非インタラクティブ モードで実行できます。Python スクリプトは、ブートフラッシュまたは揮発性スキームの下に配置する必要があります。Python CLI コマンドでは、Python スクリプトで最大 32 のコマンド ライン引数を使用できます。

Cisco MDS 9000 デバイスは、Python スクリプトを実行するためのソース CLI コマンドもサポートしています。`bootflash:scripts` ディレクトリは、ソース CLI コマンドのデフォルトのスクリプト ディレクトリです。

（注） Python3 インタープリタを使用して Python スクリプトを実行するには、スクリプトの最初の行に #!/isan/bin/python3 または #!/usr/bin/env python3 のような python3 文字列が含まれていることを確認します。

次に、スクリプトとそれを実行する方法の例を示します。

```
switch# show file bootflash:flashCheck.py
#!/bin/env python
import re
import json
import cli
import syslog

threshold = 60
ignore_paths = ['bootflash']

allmodules = json.loads(cli.clid("show module"))['TABLE_modinfo']['ROW_modinfo']
if type(allmodules) is dict:
        allmodules = [allmodules]
for eachmodule in allmodules:
        mod = eachmodule['mod']
       modtype = eachmodule['modtype']
        cmd = "slot " + str(mod) + " show system internal flash"
        if 'Supervisor' in modtype:
                s = "Supervisor(Module " + str(mod) + ")"
                regex_to_match =
r'(?P<mnton>\S+)\s+(?P<onekblks>\d+)\s+(?P<used>\d+)\s+(?P<avail>\d+)\s+(?P<useper>\d+)\s+(?P<fs>\S+)'

        elif 'Sup' in modtype:
            cmd = " show system internal flash"
            s = "Sup and LC - Module 1"
            regex_to_match =
r'(?P<mnton>\S+)\s+(?P<onekblks>\d+)\s+(?P<used>\d+)\s+(?P<avail>\d+)\s+(?P<useper>\d+)\s+(?P<fs>\S+)'
```

```
        else:
                s = "Module " + str(mod)
                regex_to_match =
r'(?P<fs>\S+)\s+(?P<onekblks>\d+)\s+(?P<used>\d+)\s+(?P<avail>\d+)\s+(?P<useper>\d+)%\s+(?P<mnton>\S+)'


        out = cli.cli(cmd)
        alllines = out.splitlines()
        for eachline in alllines:
                match = re.search(regex_to_match,eachline)
                if match:
                        grps = match.groupdict()
                        #print(grps)
                       sysstr = "Flash usage exceeds threshold({}%) on {} - Filesystem:
 {}, Mounted-On: {}, Usage:
{}%".format(str(threshold),s,grps['fs'],grps['mnton'],grps['useper'])
                        for each_ignore_path in ignore_paths:
                                if each_ignore_path in grps['mnton']:
                                        break
                                else:
                                        if int(grps['useper']) > threshold:
                                                syslog.syslog(2,sysstr)
```

Embedded Event Manager でのスクリプトの実行

Cisco MDS 9000 デバイスのEmbedded Event Manager（EEM：組み込みイベント マネージャ）のポリシーは、Python スクリプトをサポートします。

次の例は、EEM アクションとして Python スクリプトを実行する方法を示しています。

- アクション コマンドを使用することで、EEM アプレットに Python スクリプトを含めることができます。

```
switch# show running-config eem

!Command: show running-config eem
!Time: Sun May  1 14:40:07 2011

version 6.1(2)I2(1)
event manager applet a1
  event cli match "show clock"
  action 1 cli python bootflash:pydate.py
  action 2 event-default
```

- **show file** *logflash*:*event_archive_1* コマンドを実行して、ログ ファイル内のイベントによってトリガーされたアクションを検索できます。

```
switch# show file logflash:event_archive_1 | last 33

eem_event_time:05/01/2011,19:40:28 event_type:cli event_id:8 slot:active(1)
vdc:1 severity:minor applets:a1
eem_param_info:command = "exshow clock"
Starting with policy a1
Python

2011-05-01 19:40:28.644891
Executing the following commands succeeded:
          python bootflash:pydate.py
```

```
PC_VSH_CMD_TLV(7679) with q
```

Cisco MDS NX-OS のセキュリティと Python

Cisco MDS NX-OS 情報技術は、ソフトウェアの Cisco MDS NX-OS サンドボックス レイヤおよび CLI ロールベース アクセス コントロール（RBAC）によって保護されます。

Cisco MDS NX-OS network-admin または dev-ops ロールに関連付けられているすべてのユーザーは、特権ユーザーです。カスタム ロールで Python へのアクセスが許可されているユーザーは、非特権ユーザーと見なされます。非特権ユーザーは、ファイル システム、ゲスト シェル、Bash コマンドなどの Cisco MDS NX-OS 情報技術へのアクセスが制限されています。特権ユーザーは、Cisco MDS NX-OS のすべての情報技術へのより大きなアクセス権を持ちます。

セキュリティとユーザー権限の例

次の例は、特権ユーザーがコマンドを実行する方法を示しています：

```
switch# python
Python 2.7.5 (default, Oct  8 2013, 23:59:43)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('whoami')
admin
0
>>> f=open('/tmp/test','w')
>>> f.write('hello from python')
>>> f.close()
>>> r=open('/tmp/test','r')
>>> print r.read()
hello from python
>>> r.close()
```

次の例は、アクセスを拒否されている非特権ユーザーを示しています：

```
switch# python
Python 2.7.5 (default, Oct  8 2013, 23:59:43)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.system('whoami')
system(whoami): rejected!
-1
>>> f=open('/tmp/test','r')
Permission denied. Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IOError: [Errno 13] Permission denied: '/tmp/test'
>>>
```

RBAC は、ログイン ユーザー権限に基づいて CLI アクセスを制御します。ログイン ユーザーの ID は、CLI シェルまたは Bash から呼び出される Python に与えられます。Python は、Python から呼び出されたサブプロセスにログイン ユーザーの ID を渡します。

以下は、特権ユーザーの例です：

```
>>> from cli import *
>>> cli('show clock')
'11:28:53.845 AM UTC Sun May 08 2011\n'
>>> cli('configure terminal ; vrf context myvrf')
''
>>> clip('show running-config l3vm')

!Command: show running-config l3vm
!Time: Sun May  8 11:29:40 2011

version 6.1(2)I2(1)

interface Ethernet1/48
  vrf member blue

interface mgmt0
  vrf member management
vrf context blue
vrf context management
vrf context myvrf
```

以下は、非特権ユーザーの例です：

```
>>> from cli import *
>>> cli('show clock')
'11:18:47.482 AM UTC Sun May 08 2011\n'
>>> cli('configure terminal ; vrf context myvrf2')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/isan/python/scripts/cli.py", line 20, in cli
    raise cmd_exec_error(msg)
errors.cmd_exec_error: '% Permission denied for the role\n\nCmd exec error.\n'
```

次の例は、RBAC 構成を示しています：

```
switch# show user-account
user:admin
        this user account has no expiry date
        roles:network-admin
user:pyuser
        this user account has no expiry date
        roles:network-operator python-role
switch# show role name python-role
```

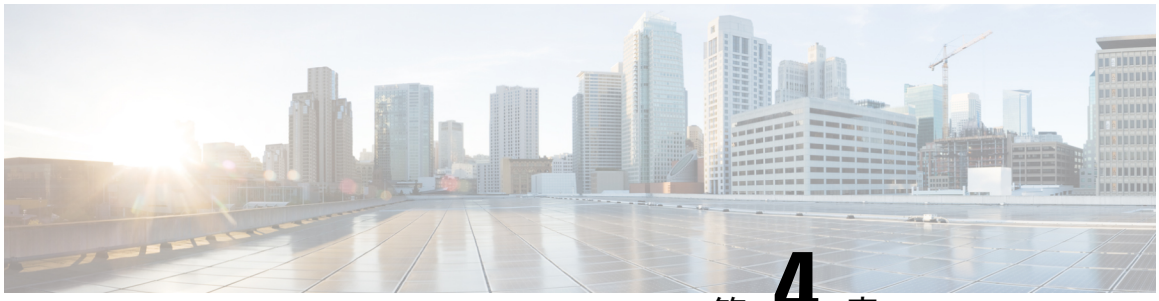
スケジューラでスクリプトを実行する例

次の例は、スケジューラ機能を使用してスクリプトを実行する Python スクリプトを示しています。

```
#!/bin/env python
from cli import *
from nxos import *
import os

switchname = cli("show switchname")
try:
    user = os.environ['USER']
except:
    user = "No user"
    pass
```

```
msg = user + " ran " + __file__ + " on : " + switchname
print msg
py_syslog(1, msg)
# Save this script in bootflash:///scripts
```

```
switch# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# feature scheduler
switch(config)# scheduler job name testplan
switch(config-job)# python bootflash:///scripts/testplan.py
switch(config-job)# exit
switch(config)# scheduler schedule name testplan
switch(config-schedule)# job name testplan
switch(config-schedule)# time start now repeat 0:0:4
Schedule starts from Mon Mar 14 16:40:03 2011
switch(config-schedule)# end
switch# term mon
2011 Mar 14 16:38:03 switch %VSHD-5-VSHD_SYSLOG_CONFIG_I: Configured from vty by admin
on 10.19.68.246@pts/2
switch# show scheduler schedule
Schedule Name       : testplan
------------------------------
User Name           : admin
Schedule Type       : Run every 0 Days 0 Hrs 4 Mins
Start Time          : Mon Mar 14 16:40:03 2011
Last Execution Time : Yet to be executed
------------------------------------------------
     Job Name            Last Execution Status
------------------------------------------------
    testplan                              -NA-
================================================================================
switch#
switch# 2011 Mar 14 16:40:04 switch  %USER-1-SYSTEM_MSG: No user ran
/bootflash/scripts/testplan.py on : switch   - nxpython
2011 Mar 14 16:44:04 switch last message repeated 1 time
switch#
```



第 4 章

Ansible

Ansibleは、クラウドプロビジョニング、構成管理、アプリケーションの展開、サービス内オーケストレーション、およびその他の IT ニーズを自動化するオープンソースの IT 自動化エンジンです。Puppet や Chef と同様に、Ansible を使用すると、管理者はさまざまなタイプのサーバー環境の管理、自動化、およびオーケストレーションを行えます。Ansible はエージェントレスであり、デバイスを自動化するためにターゲット ノード（サーバーまたはスイッチ）にソフトウェア エージェントをインストールする必要はありません。通常、Ansible は管理対象のターゲット サーバーで SSH と Python をサポートする必要がありますが、MDS スイッチの場合、Ansible は SSH と NX-API の両方を使用するように拡張されているため、スイッチ上の Python サポートは必要ありません。Ansible プレイブックは YAML で記述されているため、自動化ジョブを読みやすい形式で記述することができます。各 Ansible プレイブック内では、さまざまな Ansible モジュールを使用できます。

次に、cisco.nxos コレクション内の MDS 固有のモジュールを示します：

- nxos_vsan (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_vsan_module.html#ansible-collections-cisco-nxos-nxos-vsan-module)
- nxos_zone_zoneset (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_zone_zoneset_module.html#ansible-collections-cisco-nxos-nxos-zone-zoneset-module)
- nxos_devicealias (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_devicealias_module.html#ansible-collections-cisco-nxos-nxos-devicealias-module)
- nxos_fc_interfaces (https://docs.ansible.com/ansible/devel/collections/cisco/nxos/nxos_fc_interfaces_module.html#ansible-collections-cisco-nxos-nxos-fc-interfaces-module)

任意のコマンドを実行するには、nxos_command モジュール（https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_command_module.html#ansible-collections-cisco-nxos-nxos-command-module）を使用します。

Cisco MDS のサポートが制限されている他のモジュールもあります。Cisco MDS との互換性については、個々のモジュールのマニュアルを参照してください：https://docs.ansible.com/ansible/latest/collections/cisco/nxos/index.html#modules

Ansible モジュールは、SSH 接続または NX-API コールを行い、リアルタイムの状態データを収集し、Cisco MDS デバイスの構成を変更します。Ansible の詳細については、Ansible の公式ドキュメントを参照してください。

（注） Cisco MDS Ansible モジュールの場合、ターゲット ノードに Python インタープリタは必要ありません。

Ansible でサポートされる Cisco MDS モジュールの詳細については、Ansible モジュールを参照してください。

はじめに

Ansible のインストールについては、公式の Anisble インストールガイド、https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html を参照してください。

ホスト ファイル

ホスト ファイルには、管理対象のデバイスがリストされます。1 つのデバイスを 1 つだけのグループに含めることも、複数のグループに含めることもできます。次のホストファイルでは、2 つのデバイス mds1 と mds2 を持つ edge と呼ばれる単一のグループが使用されています。接続は、HTTPS 接続を使用してターゲット デバイスに接続する NX-API に設定されます。ユーザー名とパスワードはホストファイルに保存されます。セキュリティを強化するために、このホストファイルは Ansible Vault を使用して暗号化できますが、ここでは使用しません。

（注） Ansibe 2.5 から、 *ansible_connection: local* は廃止されました。代わりに `ansible_connection: ansible.netcommon.network_cli` または `ansible_connection: ansible.netcommon.httpapi` を使用します。httpapi 接続プラグインは、さまざまなトグルを提供します。使用可能なすべてのオプションは、Ansible のドキュメントで指定されています。network_cli 接続プラグインは、さまざまなトグルを提供します。使用可能なすべてのオプションは、Ansible のドキュメントで指定されています。

```
$ cat /etc/ansible/hosts
[all:vars]
ansible_connection = ansible.netcommon.httpapi
ansible_httpapi_use_ssl=True
ansible_httpapi_port=8443
ansible_user=username
ansible_password=password

[edge]
```

```
mds1
mds2
```

資料

すべての Cisco MDS NX-OS モジュールのマニュアルは、 https://docs.ansible.com/ansible/latest/collections/cisco/nxos/ から入手できます。または、組み込みのマニュアル ツールを使用して端末で表示できます。

```
$ansible-doc
```

プレイブックの例

この最初のプレイブックでは、いくつかの VSAN をプロビジョニングし、VSAN を削除します。**nxos_vsan** という Ansible モジュールを使用して、このタスクを自動化します。

以下に示すように、プレイブックは YAML で定義されています。

```
---
- name: Test that vsan module works
  gather_facts: no
  hosts:
  - mds1
  cisco.nxos.nxos_vsan:
    vsan:
    - id: 922
      interface:
      - fc1/1
      - fc1/2
      - port-channel 1
      name: vsan-SAN-A
      remove: false
      suspend: false
    - id: 923
      interface:
      - fc1/11
      - fc1/21
      - port-channel 2
      name: vsan-SAN-B
      remove: false
      suspend: true
    - id: 1923
      name: vsan-SAN-Old
      remove: true
  register: result
- debug: var=result
```

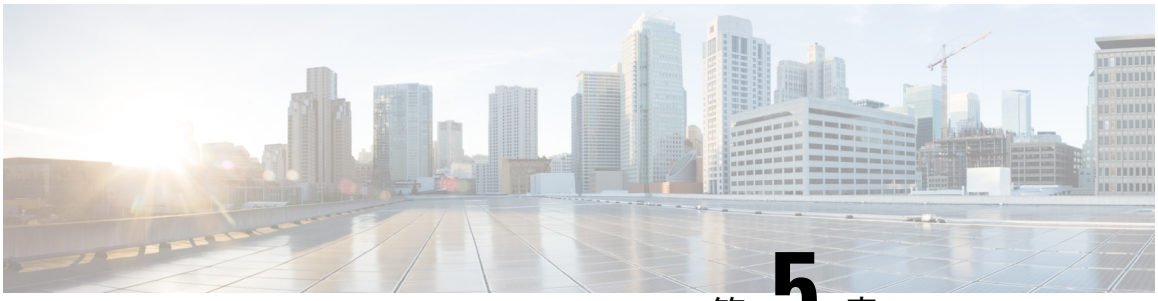
上記のプレイブックが**vsan.yml**と呼ばれると仮定すると、このタスクはターミナルから以下に示すように実行できます。

```
$ ansible-playbook vsan.yml

PLAY [VSAN TEST (NXOS)] ***************************************************

TASK [Test that vsan module works] ****************************************
changed: [mds1]
```

```
TASK [debug] ********************************************************************
ok: [mds1] => {
    "result": {
        "changed": true,
        "cmds": [
            "terminal dont-ask",
            "vsan database",
            "vsan 922",
            "vsan 922 name vsan-SAN-A",
            "no vsan 922 suspend",
            "vsan database",
            "vsan 922 interface fc1/1",
            "vsan 922 interface fc1/2",
            "vsan 922 interface port-channel 1",
            "vsan database",
            "vsan 923",
            "vsan 923 name vsan-SAN-B",
            "vsan 923 suspend",
            "vsan database",
            "vsan 923 interface fc1/11",
            "vsan 923 interface fc1/21",
            "vsan 923 interface port-channel 2",
            "vsan database",
            "no vsan 1923",
            "no terminal dont-ask"
        ],
        "failed": false,
        "messages": [
            "creating vsan 922",
            "setting vsan name to vsan-SAN-A for vsan 922",
            "no suspending the vsan 922",
            "adding interface fc1/1 to vsan 922",
            "adding interface fc1/2 to vsan 922",
            "adding interface port-channel 1 to vsan 922",
            "creating vsan 923",
            "setting vsan name to vsan-SAN-B for vsan 923",
            "suspending the vsan 923",
            "adding interface fc1/11 to vsan 923",
            "adding interface fc1/21 to vsan 923",
            "adding interface port-channel 2 to vsan 923",
            "deleting the vsan 1923"
        ]
    }
}

PLAY RECAP **********************************************************************
mds1                : ok=2    changed=1    unreachable=0    failed=0
```

まとめ

Ansible を使用して VSAN を設定/設定解除する方法を見てきました。これは単純な例ですが、自動化が必要なさまざまなタスクでモジュールを使用できます。



第 5 章

Cisco MDS SDK

Cisco MDS-SDK は、Cisco MDS スイッチ用の Python ベースのライブラリです。

このライブラリは、日常業務の自動化や、Cisco MDS スイッチが関係する新しいツールの開発に役立ちます。

Cisco MDS-SDK は NX-API を利用してデバイスと通信しますが、下位互換性のために SSH を使用することもできます。Cisco MDS NX-OS リリース 8.4(2a) 以降を実行しているスイッチには、NX-API を使用することを推奨します。

Cisco MDS-SDK の詳細については、GitHub ページ（https://github.com/Cisco-SAN/mdssdk）を参照してください。

Cisco MDS-SDK のマニュアルについては、 http://mdssdk.readthedocs.io を参照してください。

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。