NX-API CLI

NX-API CLI について

NX-API CLI は、XML 出力をサポートする Cisco NX-OS CLI システムの拡張機能です。NX-API CLI は、特定のコマンドの JSON 出力フォーマットもサポートしています。

Cisco Nexus スイッチでは、コマンドライン インターフェイス（CLI）はスイッチ上でのみ実行されます。NX-API CLI は HTTP / HTTPS を使ってスイッチの外部で CLI を使用できるようにすることで、これらの CLI のユーザー補助を改善します。この拡張機能をスイッチの既存の Cisco NX-OS CLI システムに使用できます。NX-API CLI は **show** コマンド、構成と Linux Bash をサポートします。

NX-API CLI は JSON-RPC をサポートしています。

転送

NX-APIは、転送のように HTTP または HTTPS を使用します。CLI は、HTTP / HTTPS POST 本文にエンコードされます。

NX-API は、ホスト上でネイティブに、またはゲスト シェル内で実行されるアプリケーションの、UNIX ドメイン ソケットを介してサポートされます。

NX-API バックエンドは Nginx HTTP サーバを使用します。Nginx プロセスとそのすべての子プロセスは、CPU とメモリの使用量が制限されている Linux cgroup 保護下にあります。Nginx のメモリ使用量が cgroup の制限を超えると、Nginx プロセスは再起動されて、NX-API 構成（VRF、ポート、証明書構成）が復元されます。

メッセージ形式

NX-API は、XML 出力をサポートする Cisco NX-OS CLI システムの拡張機能です。NX-API は、特定のコマンドの JSON 出力フォーマットもサポートしています。

（注）
- NX-API XML 出力は、情報を使いやすいフォーマットで表示します。
- NX-API XML は、Cisco NX-OS NETCONF 導入に直接マッピングされません。
- NX-API XML 出力は、JSON に変換できます。

セキュリティ

- NX-API は HTTPS をサポートします。HTTPS を使用すると、デバイスへのすべての通信が暗号化されます。

NX-API は、デバイスの認証システムに統合されています。ユーザーは、NX-API を介してデバイスにアクセスするための適切なアカウントを持っている必要があります。NX-API では HTTP basic 認証が使用されます。すべてのリクエストには、HTTP ヘッダーに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。

（注） ユーザーのログイン資格情報を保護するには、HTTPS の使用を検討する必要があります。

[機能（feature）] マネージャ CLI コマンドを使用して、NX-API を有効にすることができます。NX-API はデフォルトで無効になっています。

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。セッション Cookie を使用すると、デバイスに送信される後続のすべての NX-API 要求にユーザー名とパスワードが含まれます。ユーザー名とパスワードは、完全な認証プロセスの再実行をバイパスするために、セッション Cookie で使用されます。セッション 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 Cookie が必要であり、認証プロセスによって提供されます。認証プロセスの不必要な使用を避けることで、デバイスのワークロードを軽減できます。

（注） **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、調整できません。

（注） NX-API は、スイッチ上の Programmable Authentication Module（PAM）を使用して認証を行います。cookie を使用して PAM の認証数を減らし、PAM の負荷を減らします。

NX-API CLI の使用

Cisco Nexus 9000 シリーズ スイッチのコマンド、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、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。

（注） NX-API 応答コードの詳細については、NX-API 応答コードの表 （18 ページ）を参照してください。

NX-API CLI は、ローカル アクセスに対してはデフォルトで有効になっています。リモート HTTP アクセスに対してはデフォルトで無効になっています。

次の例は、NX-API CLI を構成して起動する方法を示しています。

- 管理インターフェイスを有効にします。

```
switch# conf t
Enter configuration commands, one per line.
End with CNTL/Z.
switch(config)# interface mgmt 0
switch(config-if)# ip address 10.126.67.53/25
switch(config-if)# vrf context managment
switch(config-vrf)# ip route 0.0.0.0/0 10.126.67.1
switch(config-vrf)# end
switch#
```

- NX-API **nxapi** 機能を有効にします。

```
switch# conf t
switch(config)# feature nxapi
```

次の例は、リクエストとそのレスポンスを XML 形式で示しています。

要求:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ins_api>
  <version>0.1</version>
  <type>cli_show</type>
  <chunk>0</chunk>
  <sid>session1</sid>
  <input>show switchname</input>
  <output_format>xml</output_format>
</ins_api>
```

応答：

```
<?xml version="1.0"?>
<ins_api>
  <type>cli_show</type>
  <version>0.1</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body>
```

```
        <hostname>switch</hostname>
      </body>
      <input>show switchname</input>
      <msg>Success</msg>
      <code>200</code>
    </output>
  </outputs>
</ins_api>
```

次の例は、JSON 形式の要求とその応答を示しています。

要求:

```
{
    "ins_api": {
        "version": "0.1",
        "type": "cli_show",
        "chunk": "0",
        "sid": "session1",
        "input": "show switchname",
        "output_format": "json"
    }
}
```

応答：

```
{
    "ins_api": {
        "type": "cli_show",
        "version": "0.1",
        "sid": "eoc",
        "outputs": {
            "output": {
                "body": {
                    "hostname": "switch"
                },
                "input": "show switchname",
                "msg": "Success",
                "code": "200"
            }
        }
    }
}
```

（注） ユーザーを削除しようとすると失敗し、次のようなエラー メッセージが約 12 時間ごとに表示されるという既知の問題があります。

`user delete failed for username:userdel: user username is currently logged in - securityd`

この問題は、NX-API を介してスイッチにログインしているユーザーを削除しようとした場合に発生する可能性があります。この場合、次のコマンドを入力して、最初にユーザーのログアウトを試行します。

`switch(config)# clear user username`

その後、ユーザーの削除を再試行します。回避策を試みても問題が解決しない場合は、Cisco TAC へお問い合わせください。

NX-API で権限を root にエスカレーションする

NX-API では、管理者ユーザーの権限を root アクセスの権限にエスカレーションできます。

以下は、権限をエスカレーションするためのガイドラインです：

- 特権を root にエスカレーションできるのは管理者ユーザーのみです。
- root へのエスカレーションはパスワードで保護されています。

次の例は、管理者の権限を root にエスカレーションする方法と、エスカレーションを確認する方法を示しています。root になっても、**whoami** コマンドを実行すると admin として表示されることに注意してください。ただし、admin アカウントにはすべての root 権限があります。

最初の例：

```
<?xml version="1.0"?>
<ins_api>
  <version>1.0</version>
  <type>bash</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>sudo su root ; whoami</input>
  <output_format>xml</output_format>
</ins_api>

<?xml version="1.0" encoding="UTF-8"?>
<ins_api>
  <type>bash</type>
  <version>1.0</version>
  <sid>eoc</sid>
  <outputs>
    <output>
      <body>admin </body>
      <code>200</code>
      <msg>Success</msg>
    </output>
  </outputs>
</ins_api>
```

2 番目の例：

```
<?xml version="1.0"?>
<ins_api>
  <version>1.0</version>
  <type>bash</type>
  <chunk>0</chunk>
  <sid>sid</sid>
  <input>sudo cat path_to_file </input>
  <output_format>xml</output_format>
</ins_api>

<?xml version="1.0" encoding="UTF-8"?>
<ins_api>
  <type>bash</type>
  <version>1.0</version>
  <sid>eoc</sid>
  <outputs>
    <output>
```

```
      <body>[Contents of file]</body>
      <code>200</code>
      <msg>Success</msg>
    </output>
  </outputs>
</ins_api>
```

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：Cisco Nexus 9000 NX-OS NX-API

NX-API 管理コマンド

次の表にリストされている CLI コマンドを使用して、NX-API を有効にして管理できます。

表 *1 : NX-API* 管理コマンド

NX-API 管理コマンド	説明
feature nxapi	NX-API を有効化します。
no feature nxapi	NX-API を無効化します。
nxapi {**http** | **https**} **port** *port*	ポートを指定します。
no nxapi {**http** | **https**}	HTTP / HTTPS を無効化します。
show nxapi	ポートと証明書情報を表示します。
nxapi certificate {httpscrt certfile | **httpskey keyfile**} *filename*	次のアップロードを指定します： • httpscrt が指定されている場合の HTTPS 証明書。 • httpskey が指定されている場合の HTTPS キー。 HTTPS 証明書の例： **nxapi certificate httpscrt certfile bootflash:cert.crt** HTTPS キーの例： **nxapi certificate httpskey keyfile bootflash:privkey.key**
nxapi certificate enable	証明書を有効化します。

NX-API 管理コマンド	説明
nxapi certificate sudi	この CLI は、Secure Unique Device Identifier（SUDI）を使用してデバイスを安全に認証する方法を提供します。 nginx の SUDI ベースの認証は、CISCO SUDI 準拠のコントローラによって使用されます。 SUDI は、X.509v3 証明書に含まれる IEEE 802.1AR 準拠のセキュアデバイス識別子で、Cisco デバイスの製品識別子とシリアル番号を維持します。ID は製造時に実装され、公的に識別可能なルート認証局につながれます。 （注） NX-API が SUDI 証明書を使用している場合、ブラウザ、curl などのサードパーティアプリケーションからはアクセスできません。 （注） 「nxapi certificate sudi」は、構成されるとカスタム証明書/キーを上書きします。カスタム証明書/キーを元に戻す方法はありません。 （注） 「nxapi certificate sudi」と「nxapi certificate trustpoint」と「nxapi certificate enable」は排他的であり、いずれかを構成すると他の構成は削除されます。 （注） NX-API は、SUDI 証明書ベースのクライアント証明書認証をサポートしていません。クライアント証明書認証が必要な場合は、アイデンティティ証明書を使用する必要があります。 （注） NX-API 証明書 CLI は show run の出力に存在しないため、現在の CR/ロールバックでは、「nxapi certificate sudi」オプションで上書きした場合、カスタム証明書に戻せません。
nxapi use-vrf *vrf*	デフォルト VRF、管理 VRF、または名前付き VRF を指定します。 （注） Cisco NX-OS リリース 7.0(3)I2(1) では、NGINX は 1 つの VRF でのみリッスンします。

NX-API 管理コマンド	説明
ip netns exec management iptables	アクセス制限を実装したうえで、管理 VRF で実行できます。 （注） **feature bash-shell** を有効にしてから、Bash シェルからコマンドを実行する必要があります。Bash シェルの詳細については、Bash の章を参照してください。 *Iptables* は、ポリシー チェーンを使用してトラフィックを許可または阻止するコマンドライン ファイアウォール ユーティリティであり、ほとんどの *Linux* ディストリビューションにプリインストールされています。 （注） iptables を bash シェルで変更した後、リロード後も iptables を永続化する方法の詳細については、リロード間で Iptable を永続化する（17 ページ）を参照してください。

以下は、HTTPS 証明書の正常なアップロードの例です：

```
switch(config)# nxapi certificate httpscrt certfile certificate.crt
Upload done. Please enable. Note cert and key must match.
switch(config)# nxapi certificate enable
switch(config)#
```

（注） 証明書を有効にする前に、証明書とキーを構成する必要があります。

以下は、HTTPS キーの正常なアップロードの例です：

```
switch(config)# nxapi certificate httpskey keyfile bootflash:privkey.key
Upload done. Please enable. Note cert and key must match.
switch(config)# nxapi certificate enable
switch(config)#
```

NX-API を使用したインタラクティブ コマンドの操作

対話型コマンドの確認プロンプトを無効にし、エラー コード 500 によるタイムアウトを回避するには、対話型コマンドの前に**[端末の dont-ask（terminal dont-ask）]** を追加します。 **を使用。複数の対話型コマンドを区切るには**、それぞれが。は単一のブランク文字で囲まれています。

エラー コード 500 でのタイムアウトを回避するために**端末の dont-ask** を使用する対話型コマンドの例をいくつか次に示します：

```
terminal dont-ask ; reload module 21
```

```
terminal dont-ask ; system mode maintenance
```

NX-API リクエスト要素

NX-API リクエスト要素は、XML フォーマットまたはJSON フォーマットでデバイスに送信されます。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。

次の表にリストされている NX-API 要素を使用して、CLI コマンドを指定します。

（注） ユーザーには、「configure terminal」コマンドを実行する権限が必要です。JSON-RPC が入力リクエスト フォーマットの場合、「configure terminal」コマンドは、常に、ペイロード内のコマンドが実行される前に実行されます。

表 *2 : XML* または *JSON* フォーマットの *NX-API* リクエスト要素

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	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_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** 構造化された出力が必要な CLI **show** コマンド。show コマンド専用です。**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、つまり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 • **bash** Bash コマンド。ほとんどの非対話型 Bash コマンドは、NX-API でサポートされています。 （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。 • メッセージタイプが ASCII の場合、出力でパイプ操作がサポートされます。出力が XML 形式の場合、パイプ操作はサポートされていません。 • 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。 • 対話型コマンドはサポートされていません。

NX-API リクエスト要素	説明
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。 （注） 0 チャンク出力しません。 1 チャンク出力。 （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。 • XML 出力メッセージ形式の場合（XML がデフォルトです。）< または > などの特殊文字は、有効な XML メッセージを形成するために変換されます（< は &lt に変換されます > は &gt 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。 （注）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。
ロールバック	構成 CLI に対してのみ有効であり、show コマンドに対しては有効ではありません。構成ロールバック 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：エラーを無視して他の CLI を続行します。 • Rollback-on-error：システム構成を以前の状態にロールバックします。 （注）入力リクエスト フォーマットが XML または JSON の場合、ロールバック要素は cli_conf モードで使用できます。

NX-API リクエスト要素	説明
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 メッセージタイプであり、cli_confモードではサポートされません。 （注） **bash** を除き、複数のコマンドは「;」で区切ります。（; は、単一のブランク文字で囲む必要があります。） エラーコード 500 でタイムアウトしないように、コマンドの前に `terminal dont-ask` を付加します。次に例を示します。 `terminal dont-ask ; cli_conf ; interface Eth4/1 ; no shut ; switchport` **bash** の場合、複数のコマンドは「;」で区切ります。（; は単一のブランク文字で囲まれて**いません**。） 以下は、複数のコマンドの例です。 （注） cli_show: `show version ; show interface brief ; show vlan` cli_conf: `interface Eth4/1 ; no shut ; switchport` bash: `cd /bootflash;mkdir new_dir`

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形式は次のとおりです。 （注） xml：XML 形式を指定します。 json：JSON 形式で出力を指定します。 （注） Cisco NX-OS 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 出力は出力の量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/HTTPS ヘッダーの content-type ヘッダーは、応答形式（XML または JSON）のタイプを示します。

JSON-RPC が入力リクエスト形式である場合、次の表にリストされている NX-API 要素を使用して、CLI コマンドを指定します。

表 *3 : JSON-RPC* フォーマットの *NX-API* リクエスト要素

NX-API リクエスト要素	説明
jsonrpc	JSON-RPC プロトコルのバージョンを指定する文字列。 バージョンは 2.0 であることが必要です。
method	呼び出されるメソッドの名前を含む文字列。 NX-API は、次のいずれかをサポートします。 • **cli**：show または構成コマンド • **cli_ascii**：show または構成コマンド。フォーマットせずに出力 • **cli_array**：show コマンド専用。**cli** に似ていますが、**cli_array** はデータを角括弧（[ ]）で囲まれた 1 つの要素のリスト、つまり配列として返します。

NX-API リクエスト要素	説明
params	メソッドの呼び出し中に使用されるパラメータ値を保持する構造化された値。 以下が含まれている必要があります。 • **cmd**：CLI コマンド • **version**：NX-API リクエストのバージョン識別子
ロールバック	構成 CLI に対してのみ有効であり、show コマンドに対しては有効ではありません。構成ロールバック オプション次のいずれかのオプションを指定できます。 • Stop-on-error：最初に失敗した CLI で停止します。 • Continue-on-error：失敗した CLI を無視して他の CLI を続行します。 • Rollback-on-error：システム構成を以前の状態にロールバックします。
id	クライアントによって確立されるオプションの識別子。指定されている場合は、文字列、数値、または null 値を含む必要があります。値は null にならないはずです。数値には小数部を含めません。ユーザーが id パラメータを指定しなかった場合、サーバーはリクエストが単なる通知であるとみなし、応答はしません。パラメータは*id*: 1 などのように指定します。

NX-API 応答要素

CLI コマンドに応答する NX-API 要素を次の表に示します。

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。

NX-API 応答要素	説明
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 にある場合、各コマンド出力は単一の出力タグで囲まれます。 メッセージ タイプが cli_conf または bash の場合、cli_conf および bash コマンドにはコンテキストが必要なため、すべてのコマンドに単一の出力タグがあります。
出力	単一のコマンド出力の出力を囲むタグ。 cli_conf と bash メッセージ 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原因コード。 NX-API は、ハイパーテキスト転送プロトコル（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 へのアクセスの制限

デバイスへの HTTP および HTTPS アクセスを制限するには、ACL と iptable の 2 つの方法があります。使用する方法は、`nxapi use-vrf<vrf-name>` CLI コマンドを使用して、NX-API 通信の VRF を構成していたかどうかに応じて決まります。

特定の VRF を使用するように NXAPI を構成していない場合にのみ、ACL を使用してデバイスへの HTTP または HTTPS アクセスを制限します。ACL の構成の詳細については、使用しているスイッチ ファミリの *Cisco Nexus* シリーズ *NX-OS* セキュリティ構成ガイドを参照してください。

ただし、NX-API 通信用に VRF を設定した場合、ACL は HTTP または HTTPS アクセスを制限しません。代わりに、iptable のルールを作成します。ルールの作成の詳細については、iptable の更新 （16 ページ）を参照してください。

iptable の更新

iptable を使用すると、VRF が NX-API 通信用に構成されている場合に、デバイスへの HTTP または HTTPS アクセスを制限できます。このセクションでは、既存の iptable に HTTP および HTTPS アクセスをブロックするルールを追加、確認、削除する方法を示します。

手順

ステップ 1 HTTP アクセスをブロックするルールを作成するには、次の手順を実行します。

```
bash-4.3# ip netns exec management iptables -A INPUT -p tcp  --dport 80 -j DROP
```

ステップ 2 HTTPS アクセスをブロックするルールを作成するには、次の手順を実行します。

```
bash-4.3# ip netns exec management iptables -A INPUT -p tcp  --dport 443 -j DROP
```

ステップ 3 適用されたルールを確認するには、次の手順を実行します。

```
bash-4.3# ip netns exec management iptables -L

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  anywhere             anywhere             tcp dpt:http
DROP       tcp  --  anywhere             anywhere             tcp dpt:https

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

ステップ 4 10.155.0.0/24 サブネットのポート 80 へのすべてのトラフィックをブロックするルールを作成して確認するには、次の手順を実行します。

```
bash-4.3# ip netns exec management iptables -A INPUT -s 10.155.0.0/24 -p tcp --dport 80
 -j DROP
bash-4.3# ip netns exec management iptables -L

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  10.155.0.0/24        anywhere             tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

ステップ 5 以前に適用したルールを削除して確認するには、次の手順を実行します。

この例では、最初のルールを INPUT から削除します。

```
bash-4.3# ip netns exec management iptables -D INPUT 1
bash-4.3# ip netns exec management iptables -L
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

次のタスク

iptables のルールを bash シェルで変更した場合、リロード後は保持されません。ルールを永続的にするには、リロード間で Iptable を永続化する （17 ページ）を参照してください。

リロード間で Iptable を永続化する

iptable のルールを bash シェルで変更した場合、リロード後は保持されません。このセクションでは、リロード後も変更された iptable を永続化する方法について説明します。

始める前に

iptable を変更したとします。

手順

ステップ 1 iptables_init.log という名前のファイルを /etc ディレクトリに作成します。

```
bash-4.3# touch /etc/iptables_init.log; chmod 777 /etc/iptables_init.log
```

ステップ 2 iptable の変更を保存する /etc/sys/iptables ファイルを作成します。

```
bash-4.3# ip netns exec management iptables-save > /etc/sysconfig/iptables
```

ステップ 3 次の一連のコマンドを使用して、/etc/init.dディレクトリに「iptables_init」という起動スクリプトを作成します。

```
#!/bin/sh

### BEGIN INIT INFO

# Provides:          iptables_init

# Required-Start:

# Required-Stop:

# Default-Start:     2 3 4 5

# Default-Stop:

# Short-Description: init for iptables
```

```
# Description:       sets config for iptables

#                    during boot time

### END INIT INFO


PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
start_script() {
    ip netns exec management iptables-restore < /etc/sysconfig/iptables
    ip netns exec management iptables
    echo "iptables init script executed" > /etc/iptables_init.log
}
case "$1" in
  start)
    start_script
   ;;
  stop)
    ;;
  restart)
    sleep 1
    $0 start
    ;;
  *)
    echo "Usage: $0 {start|stop|status|restart}"
    exit 1
esac
exit 0
```

ステップ 4 起動スクリプトに適切な権限を設定します。

```
bash-4.3# chmod 777 /etc/init.d/iptables_int
```

ステップ 5 chkconfig ユーティリティを使用して、「iptables_int」起動スクリプトを「オン」に設定します。

```
bash-4.3# chkconfig iptables_init on
```

「iptables_init」起動スクリプトは、リロードを実行するたびに実行されます。これで iptable ルールを永続的にすることができました。

NX-API 応答コードの表

次に、NX-API 応答の考えられる NX-API エラー、エラー コード、およびメッセージを示します。

（注） 標準の HTTP エラー コードは、ハイパーテキスト転送プロトコル (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) にあります。

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。
BASH_CMD_ERR	400	Bash コマンド エラー
CHUNK_ALLOW_ONE_CMD_ERR	400	チャンクは、1 つのコマンドだけを受け入れます。
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	着信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	構成モードは **[表示（show）]** を許可しません。
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。これは要求フォーマットのエラーです。
LIBXML_PARSE_ERR	500	システムの内部 LIBXML 解析エラー。これは要求フォーマットのエラーです。
LIBXML_PATH_CTX_ERR	500	システムの内部 LIBXML パス コンテキスト エラー。これは要求フォーマットのエラーです。
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。
BASH_CMD_NOT_SUPPORTED_ERR	501	Bash コマンドはサポートされていません。
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	大量の出力の可能性があるため、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_XML_NOT_ALLOWED_IN_INPUT	501	このコマンドへのパイプ XML は入力では許可されていません。
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	不明なエラー。

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。