NETCONF エージェント

この章は次のトピックで構成されています。

NETCONF エージェントについて

NETCONF（Network Configuration Protocol、ネットワーク構成プロトコル）は、RFC 6241 によって定義されているネットワーク管理プロトコルです。Cisco NX-OS は、クライアント側のインターフェイスである NETCONF エージェントを提供しており、XML でエンコードされた YANG モデルの形式で、クライアントの要求とサーバーの応答のため、SSH 上のセキュアな転送を提供します。

NETCONF は、構成データストアと、これらのデータストアでの操作とクエリを可能にする一連の作成、読み取り、更新、および削除（CRUD）操作を定義しています。NX-OS では、実行、起動、候補の 3 つのデータストアがサポートされています。サポートされている操作の簡単な説明を次に示します。

表 *1*: サポートされる操作

動作	説明
get	実行構成と動作状態を取得します。

動作	説明
get-config	指定されたデータストアから構成を取得します。
edit-config	指定されたターゲット データストアに指定された構成をロードします
close-session	セッションの適切な終了を要求します。
kill-session	セッションを強制終了します。
copy-config	別のデータストアの内容でデータストアを作成するか、置き換えます。
ロック	データストアをロックします。
unlock	データストアのロックを解除します。
検証	指定された構成の内容を検証します。
commit	候補構成を新しい現行の実行構成としてコミットします。
cancel-commit	進行中の要確認コミットをキャンセルします。
discard-changes	候補構成を現行の実行構成に戻します。

NETCONF に関する注意事項と制限事項

NETCONF エージェントには、次の注意事項と制限事項があります：

- Cisco NX-OS は、NETCONF 通知で Cisco デバイス YANG モデルと OpenConfig モデルの両方をサポートします。
- デバイス YANG モデルはエフェメラルデータを定義します。これらは「// Ephemeral data」というコメントでマークされます。これらの非永続的な大容量データは、モデルの残りの部分とは異なる方法で処理されます。これらは、<get> クエリの <filter> パラメータが、コメントでマークされた特定の要素を具体的に指している場合にのみ返されます。使用方法の詳細については、エフェメラルデータ サポートのドキュメントを参照してください。
- Cisco NX-OS リリース 9.3（3）以降、NETCONF は RFC 6241 に準拠していますが、次の例外があります：
  - 兄弟格納ファイルマッチ ノードは、「AND」式ではなく「OR」式で論理的に結合されます。（セクション 6.2.5）
  - 候補データストアを編集した後は、同じプロパティの実行構成を編集しないでください。

- 1 つの Get 要求でサポートされるオブジェクトの数は 250,000 です。次のエラーが表示された場合は、要求されたデータが 250,000 を超えていることを意味します。このエラーを回避するには、より狭い範囲のデータをクエリするフィルタを使用してリクエストを送信します。

```
too many objects(459134 > 250000) to query the entire device model.
```

- NETCONF は、RFC 6536 で指定されている拡張ロールベース アクセス コントロール（RBAC）をサポートしていません。「network-admin」ロールを持つユーザーのみが NETCONF エージェントへのアクセスを許可されます。

- NX-OS 9.3(1) 以降、NETCONF クライアントからスイッチへのNETCONF `get` および `get-config` 要求には、明示的な名前空間とフィルタが含まれている必要があります。この要件は、OpenConfig YANG および NETCONF デバイス モデルへの要求に影響します。次のようなメッセージが表示された場合、要求は名前空間を伝送していません。

```
Request without namespace and filter is an unsupported operation
```

次の例は、この変更前の動作を使用した `get` 要求と応答を示しています。この例は、サポートされなくなった動作が原因で発生するエラーメッセージを示しています。

要求:

```
<get>
</get>
```

応答：

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <rpc-error>
      <error-type>protocol</error-type>
      <error-tag>operation-not-supported</error-tag>
      <error-severity>error</error-severity>
      <error-message xml:lang="en">Request without filtering is an unsupported
operation</error-message>
  </rpc-error>
</rpc-reply>
```

次に、NX-OS リリース 9.3(1) 以降での正しい動作の `get` 要求と応答の例を示します。

要求:

```
<get>
  <filter>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
    </System>
  </filter>
</get>
```

応答：

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <System> …
    </data>
</rpc-reply>
```

- <edit-config> の「置換」操作は、影響を受けるシステム コンポーネントによって実装されている実行時デフォルト値と動作が原因で、機能しない場合があります。したがって、NX-API 開発者サンドボックスの代わりに、<get-config> クエリによって取得した構成上で、置換を行うための構成を基礎とする方が適切です。
- Cisco NX-OS NETCONF サーバーは、最大 5 つのサブスクリプション（クライアント セッションごとに 1 つのサブスクリプション）をサポートします。
- RFC 5277 に従って、自律通知は、イベント送信元の NETCONF、SYSLOG、および SNMP ストリームをサポートします。このリリースでは、Cisco NX-OS は NETCONF ストリームのみをサポートします。
- Cisco NX-OS は、サブスクリプションの [再生（Replay）] オプションをサポートしていません。[開始時刻（Start Time）] オプションと [終了時刻（Stop Time）] オプションは再生の一部であるため、サポートされていません。
- ストリーム サブスクリプションとフィルタリングでは、サブツリー フィルタリングのみがサポートされます。XPath フィルタリングはサポートされていません。
- Cisco NX-OS NETCONF エージェントが高負荷で動作している場合、一部のイベント通知がドロップされる可能性があります。
- Cisco NX-OS は、Cisco NX-OS リリース 9.3(1) 以降で NETCONF 通知をサポートします。Cisco NX-OS は、Cisco デバイス YANG モデルのみをサポートします。
- NATがすでに構成されている場合は、スイッチにNETCONFを構成できません。NETCONFとNATの構成には互換性がなく、同じスイッチ上で共存できません。
- Cisco NX-OS は、Cisco Device YANG モデルと OpenConfig モデルの両方をサポートします。NETCONF 通知で OpenConfig モデルをサポートするのは、Cisco NX-OS 9.3(5) リリース以降です。
- 10.2(1)F リリース以降では、操作チェックポイント、ロールバック、インストール、CA 証明書のインポート、モジュールのリロード、個々のモジュールのリロード、およびファイルのコピーがサポートされています。
- L2 MAC リーフ プロパティ値を入力として openconfig-acl NETCONF GET 操作を実行する場合は、MAC アドレスの文字を大文字フォーマット（AA:AA: AA:AA:AA:AA）で入力することをお勧めします。たとえば、source-mac: 0A:0B:0C:0D:0E:0F です。
- Cisco NX-OS リリース 10.3(1)F 以降、NETCONF は、Cisco Nexus 9808 プラットフォーム スイッチでサポートされています。

NETCONF エージェントの構成

Cisco NX-OS 9.3(5) 以降の SSH を介した NETCONF エージェントの設定

この手順では、SSHを介してNETCONFエージェントを有効にして構成する方法について説明します。

（注） この手順は、Cisco NX-OS リリース 9.3(5) 以降で使用します。

始める前に

NETCONF を使用してスイッチと通信する前に、NETCONF エージェントを有効にする必要があります。NETCONF エージェントを有効または無効にするには、**[no] feature netconf** コマンドを入力します。

手順の概要

1. **configure terminal**
2. **feature netconf**
3. （任意） **netconf idle-timeout** *it-num*
4. （任意） **netconf sessions** *num-sessions*

手順の詳細

手順

	コマンドまたはアクション	目的
ステップ 1	**configure terminal** 例： `switch# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 2	**feature netconf** 例： `switch(config)# feature netconf`	NETCONF サービスを有効にします。
ステップ 3	（任意） **netconf idle-timeout** *it-num* 例： `switch(config)# netconf idle-timeout 5`	（オプション）アイドル状態のクライアント セッションが切断されるまでのタイムアウトを分単位で指定します。*it-num* の範囲は 0 ～ 1440 分です。デフォルトのタイムアウトは 5 分です。値を 0 に設定するとタイムアウトが無効になります。

	コマンドまたはアクション	目的
ステップ 4	（任意） **netconf sessions** *num-sessions* 例： `switch(config)# netconf sessions 5`	同時クライアント セッションの最大数を指定します。*num-sessions* の範囲は 1 ～ 10 です。デフォルト値は 5 セッションです。

Cisco NX-OS 9.3(4) 以前の NETCONF エージェントの構成

（注） Cisco NX-OS リリース9.3(4) 以前の場合は、次の手順に従ってください。

NETCONFエージェントは、構成ファイル（/etc/mtx.conf）の [netconf] セクションで、次のオプションの構成パラメータをサポートします。

パラメータ	説明
idle_timeout	（オプション）アイドル状態のクライアント セッションが切断されるまでのタイムアウトを分単位で指定します。 デフォルト値は 5 分です。 値を 0 に設定するとタイムアウトが無効になります。
limit	（オプション）同時クライアント セッションの最大数を指定します。 デフォルト値は 5 セッションです。 指定できる範囲は、1 ～ 10 です。

次に、構成ファイルの [netconf] セクションの例を示します。

```
[netconf]
mtxadapter=/opt/mtx/lib/libmtxadapternetconf.1.0.1.so
idle_timeout=10
limit=1
```

変更した構成ファイルを有効にするには、CLI コマンド **[no] feature netconf** を使用して NETCONF エージェントを無効にしてから再度有効にして、再起動する必要があります。

NETCONF セッションの確立

NETCONFは、クライアントとサーバー間の永続的な接続を必要とする接続指向のプロトコルです。スイッチ上の NETCONF エージェントは、管理ポート IP アドレスのポート 830 でリッ

スンします。クライアントは、SSH を介して NETCONF サブシステムとの接続を確立できます。クライアントが NETCONF エージェントとのセッションを確立すると、サーバーは<hello>メッセージをクライアントに送信します。同様に、クライアントは<hello>メッセージをサーバーに送信します。は、<hello>メッセージは、接続が開くとすぐに同時に交換されます。各<hello>メッセージには、送信側ピアのプロトコル バージョンと機能のリストが含まれています。これらのメッセージは、プロトコルの互換性と機能を判断するために使用されます。両方の NETCONF ピアは、共通のプロトコルバージョンが他のピアの<hello>メッセージを表示します。また、サーバーの<hello>メッセージには<session-id>一方、クライアントのメッセージを送信することはできません。

次に、ssh コマンドを使用したセッション確立の例を示します。最初の<hello>メッセージがサーバーから受信され、2 番目のメッセージがクライアントから送信されます。サーバーの<hello>メッセージに、プロトコル バージョン「urn:ietf:params:netconf:base:1.1」と、Cisco NX-OS リリース 9.3(4) でサポートされている NETCONF 基本機能が表示されます。また、サーバーの<hello>メッセージには、サポートされているデータモデルが含まれています。現在の Cisco NX-OS リリースでサポートされているモデルと一致しない場合があります。

（注） サーバーの<hello>メッセージには、<session-id>ですが、クライアントのメッセージはそうではありません。

```
client-host % ssh admin@172.19.193.166 -p 830 -s netconf
User Access Verification
Password:
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
        <capability>urn:ietf:params:netconf:base:1.1</capability>
       <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>

       <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>

        <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
       <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>

        <capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>

<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=report-all</capability>


<capability>http://cisco.com/ns/yang/cisco-nx-os-device?revision=2020-04-20&module=Cisco-NX-OS-device</capability>


<capability>http://openconfig.net/yang/acl?revision=2019-11-27&module=openconfig-acl&deviations=cisco-nx-openconfig-acl-deviations</capability>


<capability>http://openconfig.net/yang/bfd?revision=2019-10-25&module=openconfig-bfd&deviations=cisco-nx-openconfig-bfd-deviations</capability>

        ...
    </capabilities>
    <session-id>1286775422</session-id>
</hello>
]]>]]><hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.1</capability>
  </capabilities>
</hello>
]]>]]>
```

コマンドで NETCONF を使用することは便利ではなく、エラーが発生しやすくなります。**ssh** 上記の例では、説明のみを目的として コマンドが使用されています。**ssh** コマンドよりも推奨される NETCONF 用に作成されたさまざまなクライアントがあります。**ssh** ncclient はそのような例の 1 つであり、「使用例」セクションで使用されています。

NETCONF は、セッションを終了するための 2 つの操作をサポートしています。<close-session> および<kill-session>。サーバーが<close-session>要求があった場合、セッションに関連付けられているロックとリソースを解放し、クライアントとの接続を閉じることで、セッションを正常に終了します。次に、<close-session>成功の要求と応答：

```
<rpc message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <close-session/>
</rpc>

<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

は、<kill-session>要求は別のセッションの終了を強制し、<session-id>要求メッセージで。を受信すると、<kill-session>要求がある場合、サーバーは現在の操作を終了し、ロックとリソースを解放し、指定されたセッション ID に関連付けられている接続を閉じます。次に、<kill-session> 成功の要求と応答：

```
<rpc message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>296324181</session-id>
  </kill-session>
</rpc>

<rpc-reply message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

に加えて、<close-session>および<kill-session>クライアントが一定時間要求を送信しない場合、セッションは自動的に終了します。デフォルトは 5 分です。アイドルタイムアウトの設定については、「NETCONF エージェントの設定」を参照してください。

NETCONF の読み取りおよび書き込み構成

このセクションでは、データストアを操作およびクエリするためにサポートされている基本プロトコル操作について説明します。クライアントは、NETCONF エージェントとのセッションを確立した後、これらの操作の RPC メッセージを送信できます。これらの操作の詳細については、基本的な使用方法の説明が記載されており、RFC 6242 を参照できます。

<get-config>

この操作により、指定したデータストアの構成データを取得します。サポートされるパラメータは次のとおりです。<filter>。<source>は、クエリ対象のデータストアを指定します。<running/>。現在アクティブな設定を保持します。<source>は、<filter>取得する指定されたデータストアの部分を指定します。

次に例を示します。要求および応答メッセージ。

- 全体を取得します。<System>サブツリー：

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
        <filter>
            <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device"/>
        </filter>
    </get-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <data>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
        ...
        </System>
    </data>
</rpc-reply>
```

- 特定のリスト項目を取得します。

```
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
        <filter>
            <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                <bgp-items>
                    <inst-items>
                        <dom-items>
                            <Dom-list>
                                <name>default</name>
                            </Dom-list>
                        </dom-items>
                    </inst-items>
                </bgp-items>
            </System>
        </filter>
    </get-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102">
    <data>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <bgp-items>
                <inst-items>
                    <dom-items>
                        <Dom-list>
                            <name>default</name>
                            ...
```

```
                                <rtctrl-items>
                                    <enforceFirstAs>enabled</enforceFirstAs>
                                    <fibAccelerate>disabled</fibAccelerate>
                                    <logNeighborChanges>enabled</logNeighborChanges>
                                    <supprRt>enabled</supprRt>
                                </rtctrl-items>
                                <rtrId>1.2.3.4</rtrId>
                            </Dom-list>
                        </dom-items>
                    </inst-items>
                </bgp-items>
            </System>
        </data>
    </rpc-reply>
```

<edit-config>

この操作は、指定された設定をターゲットデータストアに書き込みます。は、<target>パラメータは、編集するデータストアを指定します。<running/>または<candidate/>。候補データストアは、変更がコミットされるまで、実行中のデータストアに影響を与えることなく操作できます。詳細については、を参照してください。<commit>セクションを参照してください。は、<config>パラメータは、ターゲットデータストアに書き込まれるモデル化されたデータを指定します。モデルは「xmlns」属性で指定されます。任意の数の要素<config>サブツリーには「operation」属性を含めることができます。要素の操作は、新しい「操作」属性によってオーバーライドされるまで、その子孫要素に継承されます。サポートされている操作は、「merge」、「replace」、「create」、「delete」、および「remove」です。「削除」操作は、設定データが存在しない場合にエラーが返されないという点で「削除」とは異なります。「operation」属性が指定されていない場合は、マージ操作がデフォルトと見なされます。デフォルトの動作は、オプションの<default-operation>パラメータには、「merge」、「replace」、または「none」があります。

次に例を示します。<edit-config>要求および応答メッセージ。

- MTU 9216と実行コンフィギュレーションの説明を使用して、「po5」という名前のポートチャネルを作成します。

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target>
            <running/>
        </target>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
            <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                <intf-items>
                    <aggr-items>
                        <AggrIf-list xc:operation="create">
                            <id>po5</id>
                            <mtu>9216</mtu>
                            <descr>port-channel 5</descr>
                        </AggrIf-list>
                    </aggr-items>
                </intf-items>
            </System>
        </config>
    </edit-config>
</rpc>
```

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
    <ok/>
</rpc-reply>
```

- ポートチャネルのすべての設定を新しい設定に置き換えます。

```
<rpc message-id="104" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target>
            <running/>
        </target>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
            <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
               <intf-items>
                    <aggr-items>
                        <AggrIf-list xc:operation="replace">
                            <id>po5</id>
                            <mtu>1500</mtu>
                            <adminSt>down</adminSt>
                        </AggrIf-list>
                    </aggr-items>
                </intf-items>
            </System>
        </config>
    </edit-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="104">
    <ok/>
</rpc-reply>
```

- ポートチャネルを削除します：

```
<rpc message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target>
            <running/>
        </target>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
           <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                <intf-items>
                    <aggr-items>
                        <AggrIf-list xc:operation="delete">
                            <id>po5</id>
                        </AggrIf-list>
                    </aggr-items>
                </intf-items>
            </System>
        </config>
    </edit-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="105">
    <ok/>
</rpc-reply>
```

<copy-config>

この操作は、ターゲットの構成データストアを、ソース構成データストア全体のコンテンツによって置き換えます。ソースデータストアとターゲットデータストアのパラメータは次のとおりです。<target>に設定します。<source>

次に例を示します。<copy-config>要求および応答メッセージ。

- 実行構成をスタートアップ構成にコピーします。

```
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <copy-config>
        <target>
            <startup/>
        </target>
        <source>
            <running/>
        </source>
    </copy-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="106">
    <ok/>
</rpc-reply>
```

- 実行構成を候補構成にコピーします。

```
<rpc message-id="107" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <copy-config>
        <target>
            <candidate/>
        </target>
        <source>
            <running/>
        </source>
    </copy-config>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="107">
    <ok/>
</rpc-reply>
```

<lock>

は、<lock>操作を使用すると、クライアントは設定データストアをロックし、他のクライアントがデータストアをロックまたは変更するのを防ぐことができます。クライアントが保持しているロックは、<unlock>セッションの操作または終了。は、<target>パラメータを使用して、ロックするデータストアを指定します。

次に例を示します。<lock>要求および応答メッセージ。

- ロックの取得に成功した場合：

```
<rpc message-id="108" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <lock>
        <target>
            <running/>
        </target>
    </lock>
```

```
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="108">
    <ok/>
</rpc-reply>
```

- 別のセッションですでに使用されているロックの取得に失敗しました。

```
<rpc message-id="109" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <lock>
        <target>
            <candidate/>
        </target>
    </lock>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="109">
    <rpc-error>
        <error-type>protocol</error-type>
        <error-tag>lock-denied</error-tag>
        <error-severity>error</error-severity>
       <error-message xml:lang="en">Lock failed, lock is already held</error-message>

        <error-info>
            <session-id>1553704357</session-id>
        </error-info>
    </rpc-error>
</rpc-reply>
```

<unlock>

<unlock> 操作は、以前に <lock> 操作によって取得した構成のロックを解除します。を発行したのと同じセッションのみ<lock>操作では、<unlock>操作を実行します。は、<target>パラメータは、ロック解除するデータストアを指定するために使用されます。

次に例を示します。<unlock>要求および応答メッセージ。

- ロック解除

```
<rpc message-id="110" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <unlock>
        <target>
            <candidate/>
        </target>
    </unlock>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="110">
    <ok/>
</rpc-reply>
```

<get>

<get>操作は、実行中の構成とデバイスの状態情報を取得します。サポートされているパラメータは <filter> です。<filter> パラメータは、実行構成の動作状態データのうち、どの部分を取得するかを指定します。

次に例を示します。要求および応答メッセージ。

- リスト項目の実行コンフィギュレーションおよび動作状態データを取得します。

```
<rpc message-id="111" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
        <filter>
            <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                <bgp-items>
                    <inst-items>
                        <dom-items>
                            <Dom-list>
                                <name>default</name>
                            </Dom-list>
                        </dom-items>
                    </inst-items>
                </bgp-items>
            </System>
        </filter>
    </get>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="111">
    <data>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <bgp-items>
                <inst-items>
                    <dom-items>
                        <Dom-list>
                            <name>default</name>
                            <always>disabled</always>
                            <bestPathIntvl>300</bestPathIntvl>
                            <clusterId>120</clusterId>

<firstPeerUpTs>2020-04-20T16:19:03.784+00:00</firstPeerUpTs>
                            <holdIntvl>180</holdIntvl>
                            <id>1</id>
                            <kaIntvl>60</kaIntvl>
                            <mode>fabric</mode>
                            <numEstPeers>0</numEstPeers>
                            <numPeers>0</numPeers>
                            <numPeersPending>0</numPeersPending>
                            <operRtrId>1.2.3.4</operRtrId>
                            <operSt>up</operSt>
                            <pfxPeerTimeout>90</pfxPeerTimeout>
                            <pfxPeerWaitTime>90</pfxPeerWaitTime>
                            <reConnIntvl>60</reConnIntvl>
                            <rtrId>1.2.3.4</rtrId>
                            <vnid>0</vnid>
                            ...
                        </Dom-list>
                    </dom-items>
                </inst-items>
            </bgp-items>
        </System>
    </data>
</rpc-reply>
```

<validate>

この操作では、候補データストアの設定内容を検証します。これは、実行中のデータストアにコミットする前に、候補データストアで行われた設定変更を検証するのに役立ちます。パラメータは、<candidate/> 。<source>

次に例を示します。<validate>要求および応答メッセージ。

- 候補データストアの内容を検証します：

```
<rpc message-id="112" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <validate>
        <source>
            <candidate/>
        </source>
    </validate>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="112">
    <ok/>
</rpc-reply>
```

<commit>

候補構成を実行構成にコミットします。パラメータのない操作は最終と見なされ、元に戻すことはできません。次の場合<commit>が発行されます。<confirmed/>パラメータを使用すると、確定されたコミットと見なされ、別のを使用しない操作パラメータを設定します。つまり、コミットの確認です。確認済みコミットでは、次の 2 つのパラメータを使用できます。<confirm-timeout>および<persist>。は、<confirm-timeout>は、確認されたコミットが元に戻されるまでの秒数です。これにより、確認コミットが発行される前の状態に実行コンフィギュレーションが復元されます。次の場合、<confirm-timeout>が指定されていない場合、デフォルトのタイムアウトは600秒です。また、セッションが終了すると、確認されたコミットは元に戻ります。は、<persist>パラメータを使用すると、セッションが終了しても確認されたコミットが保持されます。の値は、<persist>パラメータは、任意のセッションから確認されたコミットを識別するために使用され、<persist-id>後続の確認済みコミットまたは確認コミットのパラメータ。

次に例を示します。<commit>要求および応答メッセージ。

- 候補データストアの内容をコミットします。

```
<rpc message-id="113" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit/>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="113">
    <ok/>
</rpc-reply>
```

- タイムアウトで確定されたコミット：

```
<rpc message-id="114" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <confirmed/>
        <confirm-timeout>120</confirm-timeout>
    </commit>
```

```
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="114">
    <ok/>
</rpc-reply>
```

- 永続的な確認済みコミットを開始し、永続的な確認済みコミットを確認します。

```
<rpc message-id="115" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <confirmed/>
        <persist>ID1234</persist>
    </commit>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="115">
    <ok/>
</rpc-reply>

<!-- confirm the persistent confirmed-commit, from the same session or another session
 -->
<rpc message-id="116" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <commit>
        <persist-id>ID1234</persist-id>
    </commit>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="116">
    <ok/>
</rpc-reply>
```

<cancel-commit>

この操作は、進行中の確認済みコミットをキャンセルします。別のセッションからの確認済みコミットをキャンセルする必要がある場合、<persist-id>パラメータは、<persist>パラメータを設定します。

- 同じセッションから確認されたコミットをキャンセルします。

```
<rpc message-id="117" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <cancel-commit/>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="117">
    <ok/>
</rpc-reply>
```

<discard-changes>

この操作では、実行コンフィギュレーションの内容にリセットすることによって、候補コンフィギュレーションで行われたコミットされていない変更が破棄されます。パラメータは必要ありません。

次に例を示します。<discard-changes>要求および応答メッセージ。

- 候補データストアで行われた変更を破棄します。

```
<rpc message-id="118" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <discard-changes/>
</rpc>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="118">
    <ok/>
</rpc-reply>
```

NETCONF の実行

NETCONF のモデル駆動型操作について

表 *2 : NETCONF* のモデル駆動型操作について

オペレーション	NETCONF RPC	CLI
チェックポイント	checkpoint	checkpoint <name> checkpoint <file>
ロールバック	ロールバック	rollback running-config checkpoint <name> rollback running-config checkpoint <file>
インストールするもの	install_all_nxos install_add install_activate install_deactivate install_commit install_remove	すべての nxos をインストールする <image> install {add |アクティブ化 |非アクティブ化 |コミット | remove}<rpm>
暗号化証明書のインポート	import_ca_certificate	crypto ca import <trustpoint> pkcs12 <file> <passphrase>
スイッチのリロードまたはモジュールのリロード	reload	reload [タイマー<seconds>] モジュールのリロード<module number>
ファイルのコピー	copy	コピー<destination><source>

モデル駆動型操作の例

モデル駆動型操作の例

ファイル名オプションを使用したチェックポイントの作成：

```
RPC:
<rpc message-id="checkpoint-3" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <checkpoint xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <file>bootflash:my_checkpoint2</file>
  </checkpoint>
</rpc>
```

チェックポイントuisngチェックポイント名、説明を作成しています:

```
RPC:
<rpc message-id="checkpoint-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <checkpoint xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <action>create</action>
      <name>my_checkpoint1</name>
      <description>test checkpoint one</description>
  </checkpoint>
</rpc>
```

チェックポイント名を使用してチェックポイントを削除しています:

```
RPC:
<rpc message-id="delatecheckpoint-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <checkpoint xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <action>delete</action>
      <name>my_checkpoint1</name>
  </checkpoint>
</rpc>
```

ロールバック：

（注） 次のオプションタグは、アトミック、stop-at-first-failure、best-effort として使用できます。

```
<rpc message-id="rollback-cfg-option1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<rollback xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
    <name>my_checkpoint1</name>
    <option>atomic</option>
</rollback>
</rpc>
```

ファイルオプションを使用したロールバック

```
<rpc message-id="rollback-cfg1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><
<rollback xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
     <file>bootflash:my_checkpoint2</file>
</rollback>
</rpc>
```

ファイルのコピー

リモートサーバからスイッチストレージに任意のファイルをコピーします（例：ブートフラッシュ）。

Kerry tftp の場合：プロトコルがファイル転送をサポートします。

```
<rpc message-id="copy-file-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<copy xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
  <source>tftp://172.27.xxx.xxx//<file_location?/tls1-server.pfx</source>
    <destination>bootflash:</destination>
      <vrf>management</vrf>
```

```
</copy>
</rpc>
```

CA 証明書のインポート

前提条件：スイッチで my_truspoint がすでに作成されている必要があります。

```
<rpc message-id="import_ca_certificate-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<import_ca_certificate xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
    <trustpoint>my_trustpoint</trustpoint>
    <pkcs12>tls1-server.pfx</pkcs12>
    <passphrase>xxxxxx</passphrase>
</import_ca_certificate>
</rpc>
```

RPM パッケージ EXEC RPC コマンドのインストール

Install <add>

```
<rpc message-id="install-add-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<install_add xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
 <add>rpm_packagenamehere_from_bootflash</add>
</install_add>
</rpc>
```

Install <activate>

```
<rpc message-id="install-activate-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<install_activate xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
  <activate> rpm_packagenamehere_from_bootflash</activate>
</install_activate>
</rpc>
```

Install <deactivate>

```
<rpc message-id="install-deactivate-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

  <install_deactivate xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
    <deactivate>rpm_packagenamehere_from_bootflash </deactivate>
 </install_deactivate>
</rpc>
```

Install <remove>

```
<rpc message-id="rpc-install_remove-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

<install_remove xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
<remove>rpm_packagenamehere_from_bootflash </remove>
</install_remove>
</rpc>
```

すべての nx-os イメージのインストール

```
<rpc message-id="rpc-install_all_nxos-1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

<install_all_nxos xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
  <nxos>nxos.image.bin.upg</nxos>
</install_all_nxos>
</rpc>
```

モジュール番号のリロード

```
<rpc message-id="reload-module-pyld1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<reload xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
    <module>29</module>
</reload>
</rpc>
```

再読み込み（Reload）

（注） クライアントが次の RPC を要求または送信すると、exec コマンドはスイッチのリロードを実行し、それ以上 Netconf クライアントは受信しません。<ok>応答を返します。

```
<rpc message-id="563" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<reload xmlns="http://cisco.com/ns/yang/cisco-nx-os-device"/>
</rpc>
```

NETCONF 通知

NETCONF 通知について

NETCONF 通知は、NETCONF クライアントがシステムイベントに登録し、NETCONF エージェントからこれらのイベントに関する通知を受信できるメカニズムです。これらの機能は、RFC 5277 で定義されています。http://tools.ietf.org/html/rfc5277Cisco NX-OS リリース 9.3(1) 以降、RFC 5277 で説明されているように、NETCONF 通知のサポートが開始されました。http://tools.ietf.org/html/rfc5277これは、NETCONF hello メッセージでアドバタイズされるオプションの機能です。

NETCONF クライアントは、Deviceyang または OpenConfig モデルを使用して通知をサブスクライブできます。NETCONF 通知での OpenConfig モデルのサポートは、Cisco NX-OS リリース 9.3(5) から開始されます。

このサポートにより、NETCONF クライアントは次のことができます。

- イベント通知へのサブスクライブ

  各サブスクリプションは、NETCONF クライアントからのセッションを介した 1 回限りの要求です。Cisco NX-OS NETCONF エージェントが応答し、NETCONF クライアントによってセッションが明示的に閉じられるまでサブスクリプションはアクティブです。サブスクリプションは、スイッチの再起動やスイッチの NETCONF 機能の無効化などの管理アクションによって閉じることもできます。サブスクリプションは、基盤となる NETCONF セッションがアクティブである限りアクティブです。これらの登録済みフィルタに対して生成されたイベントは、通知としてクライアントに送信されます。クライアントは、システムイベントの通知に登録できます。たとえば、ポート状態の変更、ファン速度の変更、プロセスメモリの変更などです。また、有効になっている新機能などの設定イベント。

- 障害イベント通知を受信します。

  イベント通知は、スイッチの設定イベントまたは動作イベントに関する情報を含む、整形式の XML ドキュメントです。NETCONF クライアントは、サブスクリプション要求でフィルタリング基準を送信して、すべてのイベントではなくイベントのサブセットを指定できます。

- 他の操作でイベント通知をインターリーブします。

Cisco NX-OS NETCONF エージェントは、アクティブな通知サブスクリプションを持つセッションで NETCONF 要求を受信し、処理し、応答できます。

機能交換

NETCONF ハンドシェイク中に、Cisco NX-OS NETCONF サーバーは<capabilities> 要素を接続している NETCONF クライアントに送信して、サーバーが処理できる要求を示します。交換の一部として、サーバーは次の識別子を含めます。これらの識別子は、Cisco NX-OS NETCONF サーバーが通知とインターリーブの両方をサポートしていることをクライアントに通知します。

通知の機能識別子：

```
urn:ietf:params:netconf:capability:notification:1.0
```

インターリーブの機能識別子：

```
urn:ietf:params:netconf:capability:interleave:1.0
```

イベント ストリームの検出

クライアントは、NETCONF を使用して、Cisco NX-OS NETCONF サーバーのサポートされているストリームを検出できます。使用可能なすべての<streams>。Cisco NX-OS は NETCONF ストリームのみをサポートします。イベント ストリームの検出は、要求と応答のシーケンスによって行われます。

使用可能なストリームを取得する要求：

すべての NETCONF クライアントが NETCONF を送信できます。次のフィルタを使用した要求<streams>サポートされているすべてのストリームを識別します。次の例は、クライアント要求メッセージのペイロードを示しています。

```
<rpc message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get>
      <filter type="subtree">
        <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
           <streams/>
        </netconf>
      </filter>
     </get>
</rpc>
```

応答：

Cisco NX-OS NETCONF サーバーは、クライアントがサブスクライブできる使用可能なすべてのイベント ストリームで応答します。Cisco NX-OS は NETCONF ストリームのみをサポートします。

```
<rpc-reply message-id="101"
```

```
                    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <data>
        <netconf xmlns="urn:ietf:params:xml:ns:netmod:notification">
         <streams>
            <stream>
               <name>NETCONF</name>
               <description>default NETCONF event stream </description>
            </stream>
         </streams>
        </netconf>
      </data>
</rpc-reply>
```

サブスクリプションの作成

NETCONF クライアントは、<create-subscription>プロトコル動作。Cisco NX-OS NETCONF サーバーが<ok/>要素の場合、サブスクリプションはアクティブです。

同期の Get および Set 操作とは異なり、サブスクリプションは永続的な非同期操作です。サブスクリプションは、クライアントが明示的にサブスクリプションを閉じるか、セッションがオフラインになるまでアクティブなままです。たとえば、スイッチの再起動によって。

クライアントがイベント通知をサブスクライブしていたものの、オフラインになった場合には、サーバーはサブスクリプションを終了し、セッションを閉じます。

サブスクリプションが閉じられている場合、すべてのイベント通知を受信するには、NETCONF クライアントが再接続してサブスクリプションを再度作成する必要があります。

サーバーはサブスクリプションを開始しないため、<create-subscription>操作を実行します。次に、<create-subscription> NETCONF クライアントによって送信されます。

```
<create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <stream>NETCONF</stream>
    <filter xmlns:ns1="urn:ietf:params:xml:ns:netconf:base:1.0" type="subtree">
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <intf-items>
                <phys-items>
                    <PhysIf-list>
                        <id>eth1/54/1</id>
                        <phys-items>
                          <operSt/>
                        </phys-items>
                    </PhysIf-list>
                </phys-items>
            </intf-items>
        </System>
     </filter>
</create-subscription>
```

は、<create-subscription>操作は、次のオプションのいずれかをサポートします。

- <stream>クライアントがサブスクライブするイベントのストリームを指定します。ストリームを指定しなかった場合、NETCONF ストリーム内のイベントがデフォルトでクライアントに送信されます。

- <filter>これにより、イベントをフィルタリングして、ストリームで伝送されるイベントのサブセットを提供できます。

Cisco NX-OS NETCONF サーバーは、<ok>サーバーがサブスクリプションを正常に作成できる場合は、メッセージが表示されます。

次に、クライアントで受信した成功応答の例を示します。<create-subscription>サーバーに送信するように要求します。

の応答<create-subscription>がクライアントで受信されます。

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:6ff0bda6-d3f1-4288-9a7e-0f30581e4bab">
    <ok/>
</rpc-reply>
```

（注） リプレイを使用したサブスクリプションはサポートされていないため、[開始時間（Start Time）] および [終了時間（Stop Time）] オプションは使用できません。

受信通知

NETCONF クライアントがサブスクリプションを正常に作成すると、Cisco NX-OS NETCONF サーバは、スイッチ内のすべてのイベントについて、使用されたフィルタに関連するイベント通知の送信を開始します。イベント通知は、notification 要素を含む独自の XML フォーマットのドキュメントです。

次に、クライアントが DeviceYang モデルからインターフェイス operSt にサブスクライブしていて、イーサネット インターフェイスがダウンした場合の通知の例を示します。
<create-subscription> は、[サブスクリプションの作成（Creating Subscriptions）] セクションにあります。

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2020-05-05T10:22:52.260+00:00</eventTime>
    <operation>modified</operation>
    <event>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <intf-items>
                <phys-items>
                    <PhysIf-list>
                        <id>eth1/54/1</id>
                        <phys-items>
                            <operSt>down</operSt>
                        </phys-items>
                    </PhysIf-list>
                </phys-items>
            </intf-items>
        </System>
    </event>
</notification>
```

<notification> メッセージには次のフィールドが含まれます。

- <eventTime> はイベントが発生した日時を示すタイムスタンプです。
- <operation> はモデル ノードのイベントのタイプです。
- <event> はクライアントがサブスクライブしているモデル データです。

サブスクリプションの終了

サブスクリプションは、NETCONF クライアントが NETCONF メッセージのペイロードで Cisco NX-OS NETCONF サーバーに特定の操作を送信すると終了します。サブスクリプションの終了は、次のいずれかの方法で発生します。

- サブスクリプション セッションの終了。<close-session>操作は、特定のサブスクリプション セッションの NETCONF サーバーに送信されます。
- NETCONF セッションの終了。<kill-session>操作が NETCONF サーバーに送信されます。

すべてのサブスクリプションは、1 つの NETCONF セッションに関連付けられます。これは 1 対 1 の関係です。

NETCONF の例

（注） このセクションのすべての例では、ncclient python ライブラリを使用します。

ncclient を使用した Cisco NX-OS の接続

ncclient は、NETCONF クライアント用の Python ライブラリです。次に、ncclient Manager API から Cisco NX-OS への接続を確立する方法の例を示します。

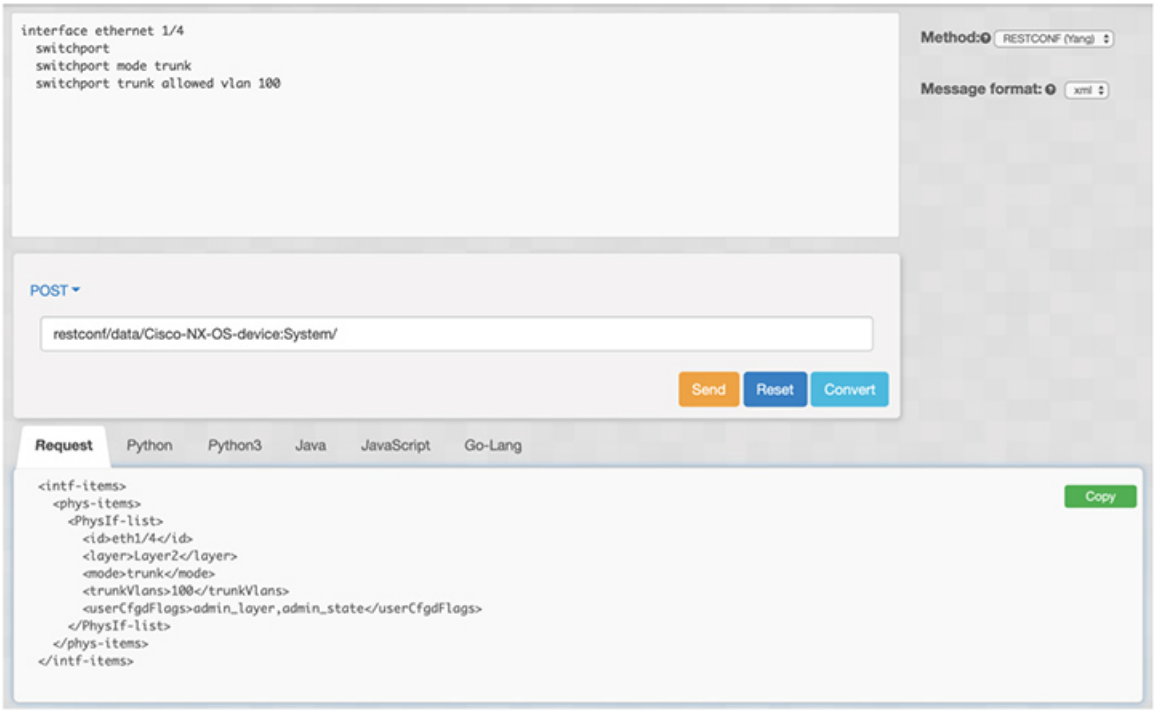
```
device = {
    "address": "10.10.10.10",
    "netconf_port": 830,
    "username": "admin",
    "password": "cisco"
}
with manager.connect(host = device["address"],
                         port = device["netconf_port"],
                         username = device["username"],
                         password = device["password"],
                         hostkey_verify = False) as m:
    # do your stuff
```

サンドボックスを使用した NETCONF ペイロードの生成

有効にするには、「NXAPI開発者サンドボックス」セクションを参照してください。NETCONF のペイロードを生成するには、メソッドを RESTCONF（Yang）に変更し、メッセージ形式を

XML に変更します。テキストウィンドウに変換する必要があるコマンドを入力し、[変換（Convert）]をクリックすると、同等のペイロードが[要求（Request）]テキストボックスに表示されます。

図 *1 : NCCLIENT*



Cisco NX-OS からの設定データの取得

次に、ncclient を使用して Cisco NX-OS から BGP 設定を取得する方法の例を示します。

```
from ncclient import manager
import sys
from lxml import etree

device = {
    "address": "nexus",
    "netconf_port": 830,
    "username": "admin",
    "password": "cisco!"
}


# create a main() method
def main():
    bgp_dom = """
    <filter type="subtree">
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
                <bgp-items>
                    <inst-items>
                        <dom-items>
                            <Dom-list/>
                        </dom-items>
                    </inst-items>
```

```
                </bgp-items>
            </System>
        </filter>
        """

        with manager.connect(host=device["address"],
                             port=device["netconf_port"],
                             username=device["username"],
                             password=device["password"],
                             hostkey_verify=False) as m:

            # Collect the NETCONF response
            netconf_response = m.get_config(source='running', filter=bgp_dom)
            # Parse the XML and print the data
            xml_data = netconf_response.data_ele
            print(etree.tostring(xml_data, pretty_print=True).decode("utf-8"))


if __name__ == '__main__':
    sys.exit(main())
```

Cisco NX-OS からの実行コンフィギュレーションおよび運用データの取得

次に、Cisco NX-OS 上のすべての物理インターフェイスのインターフェイス カウンタを取得する例を示します。

```
from ncclient import manager
import sys
from lxml import etree

device = {
    "address": "nexus",
    "netconf_port": 830,
    "username": "admin",
    "password": "cisco"
}


def main():

    intf_ctr_filter = """
    <filter>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <intf-items>
                <phys-items>
                <PhysIf-list>
                    <dbgIfIn-items/>
                    <dbgIfOut-items/>
                </PhysIf-list>
                </phys-items>
            </intf-items>
        </System>
    </filter>"""

    with manager.connect(host=device["address"],
                         port=device["netconf_port"],
                         username=device["username"],
                         password=device["password"],
                         hostkey_verify=False) as m:

        # Collect the NETCONF response
```

```
        netconf_response = m.get(filter=intf_ctr_filter)
        # Parse the XML and print the data
        xml_data = netconf_response.data_ele
        print(etree.tostring(xml_data, pretty_print=True).decode("utf-8"))

if __name__ == '__main__':
    sys.exit(main())
```

NETCONF を使用した新しい設定の作成

次に、ncclient の edit config を使用して、名前付きの VLAN 100 を作成する方法の例を示します。

```
from ncclient import manager
import sys
from lxml import etree

device = {
    "address": "nexus",
    "netconf_port": 830,
    "username": "admin",
    "password": "cisco"
}


def main():
    add_vlan = """
    <config>
      <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
        <bd-items>
          <bd-items>
            <BD-list>
              <fabEncap>vlan-100</fabEncap>
              <name>inb_mgmt</name>
            </BD-list>
          </bd-items>
        </bd-items>
      </System>
    </config>
    """


    with manager.connect(host=device["address"],
                         port=device["netconf_port"],
                         username=device["username"],
                         password=device["password"],
                         hostkey_verify=False) as m:

        # create vlan with edit_config
        netconf_response = m.edit_config(target="running", config=add_vlan)
        print(netconf_response)

if __name__ == '__main__':
    sys.exit(main())
```

NETCONF を使用した設定の削除

次に、Cisco NX-OS からループバック インターフェイスを削除する例を示します。

```
from ncclient import manager
import sys
```

```
from lxml import etree

device = {
    "address": "nexus",
    "netconf_port": 830,
    "username": "admin",
    "password": "cisco"
}


def main():
    remove_loopback = """
    <config>
      <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
        <intf-items>
          <lb-items>
            <LbRtdIf-list operation="delete">
              <id>lo10</id>
            </LbRtdIf-list>
          </lb-items>
        </intf-items>
      </System>
    </config>"""


    with manager.connect(host=device["address"],
                         port=device["netconf_port"],
                         username=device["username"],
                         password=device["password"],
                         hostkey_verify=False) as m:

        # create vlan with edit_config
        netconf_response = m.edit_config(target="running", config=remove_loopback)

        print(netconf_response)

if __name__ == '__main__':
    sys.exit(main())
```

NETCONF エージェントのトラブルシューティング

接続のトラブルシューティング

- クライアント システムから、スイッチの管理ポートに ping を実行して、スイッチが到達可能であることを確認します。
- Cisco NX-OS で、**show feature | inc netconf** コマンドを入力してエージェントのステータスを確認します。
- XML 管理インターフェイス（xmlagent とも呼ばれる）というものがあります。よく混同されますが、NETCONFエージェントとはまったく異なります。サーバーが正しいNETCONFメッセージで応答しない場合は、正しいポート830に接続していて、サーバーから正しい<hello>メッセージ（「NETCONFセッションの確立」セクションに示されているものと同様）を受信していることを確認します。

NETCONF エージェントのアカウンティング ログ

<edit-config>、<commit> または <abort> などの書き込み操作の場合、NETCONF は対応するアカウンティング ログを出力します。これには、受信した元の要求と、スイッチに適用された最終的な変更の両方が含まれます。

アカウンティングログは、**show accounting log** コマンドを使用して表示できます。

例として、次の要求を参照してください。

```
---
<edit-config>
<config xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
  <intf-items>
    <lb-items>
      <LbRtdIf-list>
        <id>lo10</id>
        <descr nc:operation="create">test</descr>
      </LbRtdIf-list>
    </lb-items>
  </intf-items>
</System>
</config>
</edit-config>
---
```

アカウンティング ログには、次の項目が含まれます。

- スイッチに適用される変更：

項目	説明
コンテキスト	セッション ID とユーザー
オペレーション	**コミット/中止**
データベース	実行または候補
ConfigMO	MO ツリーのテキスト表現。最大 3,000 文字。
ステータス	成功/失敗

例:

```
Wed Jun 29 13:48:03
2022:type=update:id=2496515744:user=admin:cmd=(COMMIT),database=[running],configMo=[
<topSystem childAction="" dn="sys" status="created,modified"><interfaceEntity
childAction=""
rn="intf" status="created,modified"><l3LbRtdIf childAction="" id="lo10" rn="lb-[lo10]"
status="created,modified"/></interfaceEntity></topSystem>] (SUCCESS)
```

- 受信した元の要求

項目	説明
コンテキスト	セッション ID とユーザー
オペレーション	**NETCONF:EDIT-CONFIG**、**NETCONF:COMMIT**、**NETCONF:ABORT**
送信元 IP（Source IP）	NETCONF クライアント IP
ペイロード	受信したXML要求。最大 3,000 文字。
ステータス	成功/失敗

例:

```
Wed Jun 29 13:48:03 2022:type=update:id=2496515744:user=admin:cmd=(NETCONF:EDIT-
CONFIG),sourceIp=[192.168.1.2],payload=[<edit-config><config
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><System
xmlns="http://cisco.com/ns/yang/cisco-nx-os-device"><intf-items><lb-items><LbRtdIf-
list><id>lo10</id><descr
nc:operation="remove">test</descr></LbRtdIf-list></lb-items></intf-
items></System></config></edit-config>] (SUCCESS)
```

失敗した要求の場合、失敗のシナリオによっては、ユーザーは両方のログを確認できない場合があります。

- 無効な要求：

  無効な要求は、構成の変更なしに拒否されるため、元の要求のみがログに記録されます。

  例:

  ```
  Wed Jun 29 20:08:36
  2022:type=update:id=2517274784:user=admin:cmd=(NETCONF:EDIT-
  CONFIG),sourceIp=[192.168.1.2],payload=[<edit-config><config
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"><System
  xmlns="http://cisco.com/ns/yang/cisco-nx-os-device"><intf-items><lb-items><LbRtdIf-
  list nc:operation="create"><id>lo10</id></LbRtdIf-list></lb-items></intf-
  items></System></config></edit-config>] (FAILED)
  ```

- さまざまな構成制限による要求の失敗：

  この場合、失敗した構成試行と元の要求の両方がログに記録されます。

  例：

  ```
  Wed Jun 29 20:11:04
  2022:type=update:id=2517274784:user=admin:cmd=(COMMIT),database=[running],
  configMo=[<topSystem childAction="" dn="sys" status="created,modified"><telemetryEntity

  childAction=""rn="tm" status="created,modified"><telemetryCertificate childAction=""

  filename="foo" hostname="foo" rn="certificate" status="created,modified"
  trustpoint="test"/></telemetryEntity></topSystem>] (FAILED)

  Wed Jun 29 20:11:04
  2022:type=update:id=2517274784:user=admin:cmd=(NETCONF:EDIT-
  CONFIG),sourceIp=[192.168.1.2],payload=[<edit-config><config
  xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"><System
  xmlns="http://cisco.com/ns/yang/cisco-nx-os-device"><tm-items><certificate-
  ```

```
items><trustpoint>test</trustpoint><hostname>foo</hostname><filename>foo</filename></certificate-
items></tm-items></System></config></edit-config>] (FAILED)
```

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。