NETCONF エージェント

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の応答のため、SSH または TLS 上のセキュアな転送を可能にします。

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。
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
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
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。
create-subscription	イベント通知ストリームへのサブスクライブ

マニュアルの変更履歴

リリース	説明（**Description**）
9.3(1)	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
9.3(3)	RFC 6241 準拠
9.3(5)	NETCONF 通知での OpenConfig モデルをサポートします
10.2(1)	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といった RPC 操作をサポートします
10.3(1)	Cisco Nexus 9800 プラットフォームスイッチ
10.3 (2)	edit-config の RBAC のサポート

リリース	説明（Description）
10.3(3)	TLS を介した Netconf トランスポートのサポート
10.4（3）	Netconf 明示モードをサポート
10.5（3）	Netconf 並列実行のサポート

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：

Netconf YANG 操作

- 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。
- Cisco NX-OS の 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）を完全にはサポートしていません。代わりに、**network-admin** 以外のロールにパスごとの EDIT 権限を付与できます。
- **network-admin** アクセス権のないユーザーは、DME 構造にアクセスできません。
- 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。次のようなエラー応答が表示された場合、要求は名前空間で伝送されていません。

  名前空間とフィルタのない要求は、サポートされていない操作です。

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

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

- `<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を、置換を行うための基礎の構成とする方が適切です。

- 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。

  デバイスモデル全体をクエリするにはオブジェクトが多すぎる（459134 > 250000）

ネイティブデバイス RPC

- Cisco NX-OS では、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といった、デバイス YANG RPC の操作がサポートされています。

サブスクリプション/通知

- Cisco NX-OS は、最大 5 つのサブスクリプションをサポートします（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）。
- 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。
- **Netconf 並列実行**の実行
  - Cisco NX-OS は、最大 10 の並列 Netconf 要求（クライアント セッションごとに 1 つの要求）をサポートします。
- Cisco NX-OS リリース 10.4(3)F 以降、NETCONF は 92348GC-X でサポートされます。

トランスポート層セキュリティ（TLS）

- Cisco NX-OS リリース 10.4（3）F 以降、TLS は、v1.3 がサポートされています。最低限サポートされるバージョンは v1.2 です。

Netconf 並列実行の実行

- Cisco NX-OS リリース 10.5(3)F 以降、最大 10 の並列 Netconf 要求（クライアントセッションごとに 1 つの要求）がサポートされます。

NETCONF エージェントの構成

SSH を介する NETCONF エージェントの構成

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。

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*

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`	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。

TLS を介する NETCONF エージェントの構成

この手順では、NETCONF エージェント over TLS を有効にして設定する方法について説明します。これにより、追加のTLS トランスポートチャンネルを SSH トランスポートと並行して実行できます。

始める前に

サーバ側とクライアント側の両方の認証に必要な証明書ファイルを準備して署名します。

（注） これは Netconf に固有のものではないため、既存のトラストポイントファイルを再利用できます。

手順の概要

1. **configure terminal**
2. （任意） **crypto ca trustpoint** *server-trustpoint*
3. （任意） **crypto ca trustpoint** *server-trustpoint* **pkcs12 bootflash:** *server-ca-file pkcs-password*
4. （任意） **crypto ca trustpoint** *client-root-trustpoint*
5. （任意） **rsakeypair** *client-key*
6. （任意） **crypto ca authenticate** *client-root-trustpoint*
7. **netconf tls certificate** *server-trustpoint*
8. **netconf tls client root certificate** *client-root-trustpoint*
9. **netconf tls port** *port*

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	（任意） **crypto ca trustpoint** *server-trustpoint* 例： `switch(config)# crypto ca trustpoint tls_server_trustpoint`	サーバ認証用のトラストポイントを作成します。 （注） 使用可能なサーバとクライアントのトラストポイントがすでにある場合、ステップ 2 ～ 6 はオプションです。

	コマンドまたはアクション	目的
ステップ 3	（任意） **crypto ca trustpoint** *server-trustpoint* **pkcs12 bootflash:** *server-ca-file pkcs-password* 例： `switch(config)# crypto ca import tls_server_trustpoint pkcs12 bootflash:server.pfx test`	サーバの pkcs12 ファイルをトラストポイントにインポートします。
ステップ 4	（任意） **crypto ca trustpoint** *client-root-trustpoint* 例： `switch(config)# crypto ca trustpoint tls_client_trustpoint`	クライアント認証用のトラストポイントを作成します。
ステップ 5	（任意） **rsakeypair** *client-key* 例： `switch(config)# rsakeypair client-key`	クライアントトラストポイントの rsa キーペアを生成します。
ステップ 6	（任意） **crypto ca authenticate** *client-root-trustpoint* 例： `switch(config)# crypto ca authenticate tls_client_trustpoint`	クライアント証明書をインポートします。この手順では、手動でコピーして貼り付ける必要があります。手順に従ってください。
ステップ 7	**netconf tls certificate** *server-trustpoint* 例： `switch(config)# netconf tls certificate tls_server_trustpoint`	サーバアイデンティティ証明書をホストするトラストポイントを入力します。
ステップ 8	**netconf tls client root certificate** *client-root-trustpoint* 例： `switch(config)# netconf tls client root certificate tls_client_trustpoint`	クライアント CA ルート証明書をホストするトラストポイントを入力します。
ステップ 9	**netconf tls port** *port* 例： `switch(config)# netconf tls port 7000`	TLS ポートを指定します。デフォルトポートは 6513 です。

キー/証明書の生成の例

次に、スイッチの bash シェルで自己署名キー/証明書を生成する例を示します。

これは実験的な使用のみであることに注意してください。アイデンティティ証明書の生成の詳細については、『*Cisco Nexus 9000* シリーズ *NX-OS* セキュリティ構成ガイド』の「**アイデンティティ証明書のインストール**」のセクションを参照してください。

（注） このタスクは、NX-OS スイッチで証明書を生成する方法の例です。任意の Linux 環境で証明書を生成することもできます。実稼働環境では、CA 署名付き証明書の使用を検討する必要があります。

1. 自己署名キーと pem ファイルを生成します。

```
switch# run bash sudo su
bash-4.3# openssl req -x509 -newkey rsa:2048 -keyout self_sign2048.key -out
self_sign2048.pem -days 365 -nodes
```

2. キー ファイルと pem ファイルを生成した後、トラストポイント CA アソシエーションで使用するためにキー ファイルと pem ファイルをバンドルする必要があります。

```
switch# run bash sudo su bash-4.3# cd /bootflash/
bash-4.3# openssl pkcs12 -export -out self_sign2048.pfx -inkey self_sign2048.key -in
 self_sign2048.pem -certfile self_sign2048.pem -password pass:Ciscolab123!
bash-4.3# exit
```

3. pkcs12 バンドルをトラストポイントに入力して、トラストポイント CA アソシエーションを設定します。

```
switch(config)# crypto ca trustpoint mytrustpoint
switch(config-trustpoint)# crypto ca import mytrustpoint pkcs12 self_sign2048.pfx
Ciscolab123!
```

4. セットアップを確認します。

```
Verify the setup.
switch(config)# show crypto ca certificates Trustpoint: mytrustpoint certificate:
subject= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
issuer= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San
Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R serial=0413
notBefore=Nov 5 16:48:58 2015 GMT notAfter=Nov 5 16:48:58 2035 GMT
SHA1 Fingerprint=2E:99:2C:CE:2F:C3:B4:EC:C7:E2:52:3A:19:A2:10:D0:54:CA:79:3E
purposes: sslserver sslclient
CA certificate 0: subject= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San
Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
issuer= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San
Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R serial=0413
notBefore=Nov 5 16:48:58 2015 GMT notAfter=Nov 5 16:48:58 2035 GMT
SHA1 Fingerprint=2E:99:2C:CE:2F:C3:B4:EC:C7:E2:52:3A:19:A2:10:D0:54:CA:79:3E
purposes: sslserver sslclient
```

NETCONF セッションの管理

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。NETCONF エージェントは、ポート 22 で実行されている XML サーバーと混同されることが多いことに注意してください。ポート 22 で実行されている XML サーバーについて

は、『 *Cisco Nexus 9000* シリーズ *NX-OS* プログラマビリティ ガイド、リリース *10.5*（*x*）』の「XML Management Interface」の章を参照してください。

セッションの開始

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、クライアントも<hello>メッセージをサーバに送信します。<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。NETCONFクライアントとサーバの両方は、相手の<hello>メッセージで、共通したプロトコルバージョンがアドバタイズされていることを確認する必要があります。また、サーバの<hello>メッセージには<session-id>を含める必要がありますが、クライアントの<hello>メッセージに<session-id>を含めてはなりません。

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>メッセージはサーバから受信され、2番目のメッセージはクライアントから送信されたものです。サーバの<hello>メッセージは、プロトコルバージョンの「urn:ietf:params:netconf:base:1.1」と、サポートされているデータモデルを示しています。次の例は、現在のCisco NX-OSリリースを反映していない可能性があります。

（注） サーバの<hello>メッセージには<session-id>が含まれていますが、クライアントのメッセージには含まれていません。

```
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>
...
    </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>
]]>]]>
```

素の **ssh** 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。RFC 6242（SSH 上で NETCONF プロトコルを使用する）から分かるとおり、メッセージの枠組みが複雑だからです。上の **ssh** コマンドは、説明のみを目的として使用しています。**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存在します。ncclient はそのような例の 1 つです。例のセクションを参照してください。

セッションの終了

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> と <kill-session> です。サーバは <close-session> 要求を受け取ると、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により、問題の生じない方法でセッションを終了します。

次の例は、<close-session> 要求と、それが正しく実行されたとの応答を示しています。

```
<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>
```

<kill-session> 要求は別のセッションを強制的に終了します。要求メッセージには <session-id> が必要です。<kill-session> 要求を受信すると、サーバ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。

次の例は、<kill-session> 要求と、それが正しく実行されたとの応答を示しています。

```
<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>
```

<close-session> と <kill-session> 要求以外にも、クライアントが一定時間何も要求を送信しなかった場合、セッションは自動的に終了されます。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 Get / Set

このセクションでは、データストアの操作とクエリのためにサポートされているNETCONF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。これらの操作の基本的な使用方法について説明しています。詳細については、RFC 6242 を参照してください。

<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 <source> と <filter> です。<source> は、<running/> のように、クエリの対象となるデータストアを指定します。これは現在アクティブな構成を保持しています。<filter>は、指定されたデータストアのうち、どの部分を取得するかを指定します。

次に、<get-config> を使用する要求および応答メッセージの例を示します。

- 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、指定された構成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/>。<running/> データストアを操作すると、スイッチの構成に直ちに変更が加えられます。一方、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。

<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。意図するモデルは「xmlns」属性で指定します。<config> サブツリーの任意の数の要素に、それぞれ 1 つの「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。「operation」属性が指定されていない場合は、「merge」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> パラメータによって上書きできます。パラメータには「merge」、「replace」、「none」などがあります。

- 「merge」操作は、構成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「create」とは異なります。
- 「remove」操作は、構成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「delete」とは異なります。

次に、<edit-config> を使用する要求および応答メッセージの例を示します。

- 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、それぞれ <source> と <target> です。

次に、<copy-config> を使用する要求および応答メッセージの例を示します。

- 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>

<lock>操作を使用すると、クライアントは構成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>セッションの操作時、またはセッションの終了時にリリースされます。<target>パラメータは、ロックするデータストアを指定します。

次に、<lock> を使用する要求および応答メッセージの例を示します。

- 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、<lock> 操作によって取得した構成のロックをリリースします。<lock> 操作を発行したのと同じセッションでのみ、<unlock> 操作を使用できます。<target> パラメータは、ロックを解除するデータストアを指定するために使用します。

次に、<unlock> を使用する要求および応答メッセージの例を示します。

- 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、<get> を使用する要求および応答メッセージの例を示します。

- 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。<source> パラメータは <candidate/> をサポートします。

次に、<validate> を使用する要求および応答メッセージの例を示します。

- 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。

<commit> は、<confirmed/> パラメータとともに発行されます。これは確認を求めるコミットと見なされ、<confirmed/> パラメータを持たない別の <commit> 操作が続く場合にのみ、実際に確定します。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>。<confirm-timeout> は、確認を求めるコミットが発行されてから、そのコミットが元に戻されて、実行構成が復元されるまでの秒数です。その間に確認を行うコミットが発行されると、コミットは確定します。または、確認を求める別のコミットにより、タイムアウトがリセットされる場合もあります。<confirm-timeout> が指定されていない場合の、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。<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を求めるコミットは保持されます。<persist> パラメータの値は、任意のセッションからの確認を求めるパラメータを識別するために使用されるもので、その後の確認を求めるコミットまたは確認を行うコミットの <persist-id> パラメータの値でも使用する必要があります。

次に、<commit> を使用する要求および応答メッセージの例を示します。

- 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を求めるコミットをキャンセルする必要がある場合、確認を求めるそのコミットの<persist> パラメータで与えたのと同じ値を、<persist-id> パラメータで使用する必要があります。

- 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、<discard-changes> を使用する要求および応答メッセージの例を示します。

- 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 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。https://datatracker.ietf.org/doc/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、デバイス YANG 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。

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ればアクティブです。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、構成または動作の変更イベントの通知にサブスクライブできます。そのうちのいくつかとして、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、機能の有効化などがあります。

- **イベント通知の受信**

  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。

- **他の動作とのイベント通知のインターリーブ**

  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 サーバが <ok/> 要素で応答した場合、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。

サーバはサブスクリプションを開始しないため、ユーザーが <create-subscription> の送信操作を含むクライアントプログラムを書く必要があります。

次に、NETCONF クライアントによる <create-subscription> の送信例を示します。

```
<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>
```

<create-subscription> 操作は、次のオプションのどれでもサポートします。

- <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 サーバは、サブスクリプションを正常に作成できた場合、応答で <ok> メッセージを送り返します。

次に、<create-subscription> を送信した後、クライアントが受信した成功の応答の例を示します。

```
<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） リプレイを使用する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。

通知を受け取る

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、リクエストされたフィルタにマッチするスイッチのすべてのイベントについて、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、インターフェイス「eth1/54/1」がダウンしたときの通知の例を示します。この場合、クライアントは上記の例のインターフェイス operSt に登録しています。

```
<?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>、イベントが発生した日時を示すタイムスタンプです。
- <operation> 、モデルノードのイベントのタイプです。
- <event>、クライアントが登録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 クライアントが次のいずれかの方法で Cisco NX-OS NETCONF サーバに特定の要求を送信すると終了します。

- 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 デバイス YANG RPC

NETCONF のモデル駆動型 RPC について

YANGは、スイッチ構成の操作に加えて、YANGモデルを介してスイッチ内の特定のアクションをトリガーする拡張性を提供します。これは、 CLIを介したEXECモードコマンドの呼び出しに相当します。Cisco NX-OSでは、次のネイティブデバイス RPC が定義されています。

操作	デバイス YANG RPC	CLI	
チェックポイント	checkpoint	**checkpoint** <*name*> **checkpoint** <*file*>	180
ロールバック	ロールバック	**rollback running-config checkpoint** <*name*> **rollback running-config checkpoint** <*file*>	6900
インストールするもの	install_all_nxos	**install all nxos** <*image*>	1800
	install_add install_activate install_deactivate install_commit install_remove	**install** {**add** | **activate** | **deactivate** | **commit** | **remove**} <*rpm*>	180
暗号化証明書のインポート	import_ca_certificate	**crypto ca import** <*trustpoint*> **pkcs12** <*file*> <*passphrase*>	5
スイッチのリロードまたはモジュールのリロード	reload	**reload** [**timer** <*seconds*>] **reload module** <*module number*>	180
ファイルのコピー	copy	**copy** <*source*> <*destination*>	7200
ファイル/ディレクトリ情報	dir	dir <*file*>	180
cli を実行	cli	<cli command>	1800

ネイティブデバイス RPC 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
  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>
```

- チェックポイント名、説明を使用したチェックポイントの作成

```
<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たチェックポイントの削除

```
<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（**Rollback**）

（注） 次のオプションタグは、atomic（アトミック）、stop-at-first-failure（最初の失敗時に停止）、best-effort（ベストエフォート）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（ブートフラッシュなど）に任意のファイルをコピーします。

```
<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

（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>
```

ディレクトリまたはファイル情報

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

スイッチで CLI コマンドを実行します

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

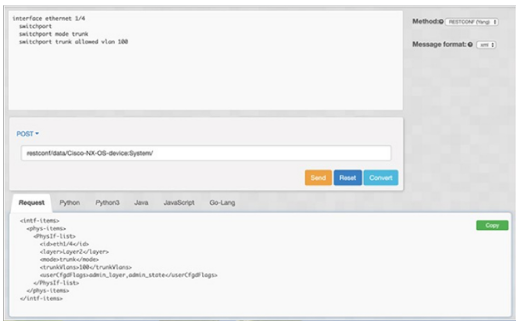
Netconf クライアントの例

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の必要なコマンドを入力し、**[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 : NX-OS* 開発者サンドボックス



ncclient を使用した Cisco NX-OS の接続

（注） さまざまなNETCONF クライアントが存在します。このセクションのすべての例では、特定の ncclient python ライブラリを使用しています。

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
```

構成データの入手

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())
```

実行構成と運用データの取得

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())
```

新しい構成の作成

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())
```

構成の削除

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

機能ステータスの確認

- Cisco NX-OS で、**show feature | inc netconf** コマンドを入力してエージェントの構成を確認します。
- NETCONF エージェントのステータスを表示するには、**show feature** コマンドを使用します。

```
switch-1# show feature | grep netconf
restconf 1 enabled
switch-1#
```

接続性の確認

- 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。
- 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。

TLS を確認

TLS が使用されている場合、ユーザーは **show netconf internal tls service statistics** コマンドを使用して、TLS のステータスを確認できます。

最初に、サーバの開始時刻を確認します。TLS サーバが実行されているかどうかがわかります。

サーバが実行されていない場合は、次に証明書とクライアントルート証明書をチェックして、証明書が有効かどうかを確認します。

```
# show netconf internal tls service statistics
==============
TLS Service
==============
Port            : 6513
Cert notBefore : Nov  5 16:48:58 2015 GMT
Cert notAfter  : Nov  5 16:48:58 2035 GMT
Client Root Cert notBefore : Oct  1 18:00:24 2020 GMT
Client Root Cert notAfter  : Sep 26 18:00:24 2040 GMT
Server restarts            : 108
Created sessions           : 54
Start                      : 02/23 01:13:10
Run time                   : 142 hour(s) 15 min(s)sec(s)
```

サーバが正常に動作しているように思われる場合は、**show netconf internal tls session all summary** コマンドを使用して、セッションのステータスをさらに確認します。

- サーバが TLC 接続をまったく受信しない場合は、クライアント側の接続をデバッグすることをお勧めします。
- サーバが実際にセッションを受信しているものの、すぐにセッションを拒否する場合は、ユーザー認証を確認することを推奨します。証明書が適切に構成されているか確認します。

```
# show netconf internal tls session all summary
N9k (config)# show netconf internal tls session all summary
==============
TLS Session
==============
* - history
Client                                        Read (KB)   Write (KB)  Status
--------------------------------------------- ----------- ----------- ------------
```

```
*10.28.23.116:35490                                    0.4         1.1 End
*10.28.23.116:35494                                    0.4         1.1 End
--                                                     0.0         0.0 Listening
```

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。これは、Netconf を経由した構成変更の履歴を確認するのに役立ちます。

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 NETCONF 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た変更

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、N
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
tem	説明
コンテキスト	セッション ID とユーザー

例:

```
Wed Jun 29 13:48:03
2022:type=update:id=2496515744:user=admin:cmd=(NETCONF:EDITCONFIG),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><LbRtdIflist><id>lo10</id>
<descr
nc:operation="remove">test</descr></LbRtdIf-list></lb-items></intfitems></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:EDITCONFIG),
  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><LbRtdIflist
   nc:operation="create"><id>lo10</id>
  </LbRtdIf-list></lb-items></intfitems></System></config></edit-config>] (FAILED)
  ```

- さまざまな構成制限による失敗：

  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:EDITCONFIG),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><certificateitems><trustpoint>test</trustpoint><hostname>foo</hostname>
<filename>foo</filename></certificateitems></tm-items></System></config></edit-config>]
 (FAILED
```

NETCONF 明示モードについて

Network Configuration（ネットワーク構成、NETCONF）プロトコルの明示モードは、RFC 6243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。このプロトコルは、NETCONFサーバからデフォルト構成を読み取るための3つの標準モードを定義します。標準モードは、**report-all**、 **trim**、および **explicit**です。

Cisco NX-OS は、**report-all** モードをすでにサポートしています。**report-all** モードを使用して構成を読み取ると、デフォルト構成を含むすべての構成がユーザーに表示されます。この機能でNETCONF クライアントは、**explicit**（明示）モードで構成を読み取ることが付加的に可能です。明示モードでは、 Cisco NX-OSは、ユーザーが明示的に行った構成のみを公開します。

Cisco NX-OS が NETCONF の明示モードでサポートするのは、**<get-config>** と **<edit-config>** RPC だけです。

- **<get-config>**

  明示モードで **<get-config>** は、その値に関係なく、ユーザーによって明示的に設定されたデータを取得します。

- **<edit-config>**

  明示モードでの作成、削除、マージ、置換、移動などの操作は、 **report-all** モードの場合とは少し異なります。

  - クライアントによってスキーマのデフォルト値に設定されたデータノードの **create** 操作属性は、有効であっても、**data-exists** エラータグで失敗します。サーバによってスキーマのデフォルト値に設定されたデータノードの有効な **create** 操作属性は、有効であれば成功します。
  - クライアントによってスキーマのデフォルト値に設定されたデータノードの **delete** 操作属性は、有効であれば成功します。サーバによってスキーマのデフォルト値に設定されたデータノードの **delete** 操作属性は、有効であっても、**data-missing** エラータグで失敗します。

注意事項と制約事項

- Cisco NX-OSは、 `get-config` および `edit-config` 以外の操作をサポートしていません。

- この機能は RFC 6243 の部分的なサポートであるため、スイッチは with-default の明示的な機能をアドバタイズしません。
- クリーンに起動されたスイッチからの明示的な get-config 応答が空になることはありません。
- この機能は、複数のスイッチが同時にスイッチにアクセスすることを許可されている場合、予期される応答を保証しません。
- 以前の 10.3(4) からのアップグレードには、次の制限が課されます。
  - インストール ... non-xx
  - インストール ..
- 任意のリリースからのリロードでは、次の制限が課されます。
- reload ascii
- 以前のリリースからのアップグレード
- 以前のリリースからのダウングレード

NETCONF 明示モードの Get/Set

明示モードを説明するために、次のテレメトリ構成モデルを考えてみましょう。

```
+--rw tm-items
  |   +--rw dest-items
  |   |  +--rw DestGroup-list* [id]
  |   |       +--rw id                   telemetry_IDType
  |   |       +--rw addr-items
  |   |       |  +--rw Dest-list* [addr port]
  |   |       |     +--rw addr      address_Ip
  |   |       |     +--rw port      uint16
  |   |       |     +--rw proto?    telemetry_Protocol      <===== Default Config
  |   |       |     +--rw enc?       telemetry_Encoding    <===== Default Config
  |   |       |     +--rw nodeid?   telemetry_NodeIDorZero
```

設定を追加

- 次の edit-config 要求は、id と、宛先アドレスおよびポートを構成します。

```
<edit-config>
    <with-defaults

xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">explicit</with-defaults>

    <target>
        <running/>
    </target>
    <config>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <tm-items>
                <dest-items>
                    <DestGroup-list>
```

```
            <id
              xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
              nc:operation="create">1</id>
                        <addr-items>
                            <Dest-list>
                                <addr>2.2.2.2</addr>
                                <port>2</port>
                            </Dest-list>
                        </addr-items>
                    </DestGroup-list>
                </dest-items>
            </tm-items>
        </System>
    </config>
</edit-config>
-----

Response:

-----

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <ok/>
</rpc-reply>
```

- 次の report-all get-config 要求は、単にすべてのデータを返します。

```
<get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <with-defaults

xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all</with-defaults>

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

 Response:

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <data>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <adminSt>enabled</adminSt>
        <batchDmeEvt>true</batchDmeEvt>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
                <enc>GPB</enc>
                <proto>gRPC</proto>
```

```
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </data>
</rpc-reply>
```

- 次の明示的な get-config 要求では、構成されたデータのみが返されます。

```
Explicit <get-config> request:

<get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <with-defaults

xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">explicit</with-defaults>

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


Response:

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <data>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </data>
</rpc-reply>
```

CLI を使用して構成を追加する

次に、明示モードで NETCONF 要求/応答を構成する手順の例を示します。

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ip address 2.2.2.2 port 2 protocol UDP encoding JSON
```

- 上記の明示的な get-config 要求は、次の応答を返します。

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
```

```
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <data>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
                <enc>JSON</enc>
                <proto>UDP</proto>
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </data>
</rpc-reply>
```

構成の削除

```
<edit-config>
  <with-defaults

xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">explicit</with-defaults>

  <target>
    <running/>
  </target>
  <config>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
                <enc
                  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
                  nc:operation="delete">JSON</enc>
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </config>
</edit-config>


Response:

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <ok/>
</rpc-reply>
```

- 上記の明示的な get-config 要求は、次の応答を返します。

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="2">
  <data>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
                <proto>UDP</proto>
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </data>
</rpc-reply>
```

CLI を使用して構成を追加する

明示モードを構成するには、次の手順を実行します。

手順の概要

1. **configure terminal**
2. **telemetry**
3. **destination-group** *dgrp_id*
4. **ip address** *ip_address* port *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol*

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`	グローバル構成モードを開始します。
ステップ 2	**telemetry** 例： `switch# telemetry`	ストリーミング テレメトリの構成モードに入ります。

	コマンドまたはアクション	目的
ステップ 3	**destination-group** *dgrp_id* 例： `switch# destination-group 1`	接続先グループを作成して、接続先グループ構成モードを開始します。
ステップ 4	**ip address** *ip_address* port *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol* 例： `switch# ip address 2.2.2.2 port 2 protocol UDP encoding JSON`	エンコードされたテレメトリ データを受信する IPv4 IP アドレスとポートを指定します。

例

- 上記の明示的な get-config 要求は、次の応答を返します。

```
-----

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
                <enc>JSON</enc>
                <proto>UDP</proto>
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </data>
</rpc-reply>

-----
```

構成の削除

```
Request:

-----
<edit-config>
    <with-defaults
xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">explicit</with-defaults>

    <target>
        <running/>
    </target>
    <config>
        <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
            <tm-items>
```

```
                <dest-items>
                    <DestGroup-list>
                        <id>1</id>
                        <addr-items>
                            <Dest-list>
                                <addr>2.2.2.2</addr>
                                <port>2</port>
                            <enc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
 nc:operation="delete">JSON</enc>
                            </Dest-list>
                        </addr-items>
                    </DestGroup-list>
                </dest-items>
            </tm-items>
        </System>
    </config>
</edit-config>

-----

Response:

-----

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <ok/>
</rpc-reply>

-----
```

- 上記の明示的な get-config 要求は、次の応答を返します。

```
-----

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <System xmlns="http://cisco.com/ns/yang/cisco-nx-os-device">
      <tm-items>
        <dest-items>
          <DestGroup-list>
            <id>1</id>
            <addr-items>
              <Dest-list>
                <addr>2.2.2.2</addr>
                <port>2</port>
                <proto>UDP</proto>
              </Dest-list>
            </addr-items>
          </DestGroup-list>
        </dest-items>
      </tm-items>
    </System>
  </data>
</rpc-reply>

-----
```

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。