XML 管理インターフェイス

- XML 管理インターフェイスについて （1 ページ）
- XML 管理インターフェイスのライセンス要件, on page 3
- XML 管理インターフェイスを使用するための前提条件, on page 3
- XML 管理インターフェイスを使用, on page 3
- サンプル XML インスタンスに関する情報 （18 ページ）
- その他の参考資料, on page 25

XML 管理インターフェイスについて

XML 管理インターフェイスについての情報

XML 管理インターフェイスを使用してデバイスを構成できます。インターフェイスは XML ベースのネットワーク構成プロトコル（NETCONF）を使用します。これにより、デバイスを管理し、インターフェイスを介して XML 管理ツールまたはプログラムと通信できます。NETCONF の Cisco NX-OS 導入では、デバイスとの通信にセキュア シェル（SSH）セッションを使用する必要があります。

NETCONF は、リモート プロシージャ コール（RPC）メッセージ内にデバイス構成要素を含めることができる XML Schema（XSD）を使用して導入されます。RPC メッセージ内から、デバイスに実行させたいコマンドのタイプに一致する NETCONF 操作の 1 つを選択します。NETCONF を使用して、デバイスで CLI コマンドのセット全体を設定できます。NETCONF の使用については、NETCONF XML インスタンスの作成, on page 7 と RFC 4741 を参照してください。

SSH を介した NETCONF の使用の詳細については、RFC 4742 を参照してください。

このセクションは、次のトピックで構成されています。

NETCONF レイヤ

次のテーブルは NETCONF レイヤのリストです：

Table 1: NETCONF レイヤ

レイヤ	例
トランスポート プロトコル	SSHv2
RPC	RPC、RPC-reply
操作	get-config、edit-config
コンテンツ	**show** または **configuration** コマンド

以下は、4 つの NETCONF レイヤの説明です。

- SSH トランスポート プロトコル：クライアントとサーバー間の暗号化接続を提供します。
- RPC タグ：リクエスタからの構成コマンドと、それに対応する XML サーバからの応答を導入します。
- NETCONF 操作タグ：構成コマンドのタイプを示します。
- 格納ファイル — 構成する機能の XML 表現を示します。

SSH xmlagent

デバイス ソフトウェアは、SSH バージョン 2 を介した NETCONF をサポートする xmlagent と呼ばれる SSH サービスを提供します。

Note xmlagent サービスは、Cisco NX-OS ソフトウェアでは XML サーバと呼ばれます。

NETCONF over SSH は、クライアントと XML サーバ間の hello メッセージの交換から始まります。最初の交換の後、クライアントは XML 要求を送信し、サーバは XML 応答で応答します。クライアントとサーバは、文字シーケンス > で要求と応答を終了します。この文字シーケンスは XML では有効ではないため、クライアントとサーバはメッセージがいつ終了するかを解釈でき、通信の同期が維持されます。

使用できる XML 構成インスタンスを定義する XML スキーマについては、NETCONF XML インスタンスの作成, on page 7 で説明されています。

XML 管理インターフェイスのライセンス要件

製品	ライセンス要件
Cisco NX-OS	XML 管理インターフェースにはライセンスは必要ありません。ライセンスパッケージに含まれていない機能は Cisxo NX-OS のイメージにバンドルされており、無料で提供されます。NX-OS ライセンス方式の詳細については、『*Cisco NX-OS Licensing Guide*』を参照してください。

XML 管理インターフェイスを使用するための前提条件

XML 管理インターフェイスを使用するには、次の前提条件があります。

- クライアント PC に SSHv2 をインストールする必要があります。
- クライアント PC に NETCONF over SSH をサポートする XML 管理ツールをインストールする必要があります。
- デバイスの XML サーバに適切なオプションを設定する必要があります。

XML 管理インターフェイスを使用

このセクションでは、XML 管理インターフェイスを手動で構成して使用する方法について説明します。

Note デバイスのデフォルト設定で XML 管理インターフェイスを使用します。

CLI による SSH および XML サーバー オプションの構成

デフォルトで SSH サーバーが有効になっています。SSH を無効にする場合は、クライアント PC で SSH セッションを開始する前に有効にする必要があります。

XML サーバー オプションを構成して、同時セッションの数とアクティブ セッションのタイムアウトを制御できます。XML ドキュメントの検証を有効にして、XML セッションを終了することもできます。

Note XML サーバ タイムアウトはアクティブ セッションだけに適用できます。

SSH の構成の詳細については、ご使用のプラットフォームの Cisco NX-OS セキュリティ構成ガイドを参照してください。

XML コマンドの詳細については、ご使用のプラットフォームの Cisco NX-OS システム マネジメント 構成ガイドを参照してください。

Procedure

Step 1 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。

configure terminal

Step 2 （オプション）XML サーバー構成およびアクティブ XML サーバー セッションに関する情報を表示します。コマンド出力でセッション番号を確認できます。

show xml server status

Step 3 指定されたサーバー セッションの XML ドキュメントを検証します。

xml server validate all

Step 4 指定した XML サーバ セッションを終了します。

xml server terminate *session*

Step 5 （オプション）キーを生成できるように SSH サーバーを無効にします。

no feature ssh

Step 6 SSH サーバをイネーブルにします。（デフォルトでは有効になっています。）

feature ssh

Step 7 （オプション）SSH サーバーのステータスを表示します。

show ssh server

Step 8 許容される XML サーバーのセッション数を設定します。

xml server max-session *sessions*

有効な範囲は 1 ～ 8 です。デフォルト値は 8 です。

Step 9 XML サーバー セッションを終了するまでに待機する秒数を設定します。

xml server timeout *seconds*

範囲は 1～1200 です。デフォルトは 1200 秒です。

Step 10 （オプション）XML サーバー設定およびアクティブ XML サーバー セッションに関する情報を表示します。

show xml server status

Step 11 （任意）実行コンフィギュレーションをスタートアップ コンフィギュレーションに保存します。

copy running-config startup-config

Example

次の例は、CLI を通して SSH と XML サーバー オプションを構成する方法の例を示しています：

```
switch# configure terminal
switch(config)# xml server validate all
switch(config)# xml server terminate 8665
switch(config)# no feature ssh
switch(config)# feature ssh server
switch(config)# xml server max-session 6
switch(config)# xml server timeout 1200
switch(config)# copy running-config startup-config
```

SSHv2 セッションの開始

クライアント PC で SSHv2 セッションを開始するには、次のような **ssh2** コマンドを使用します。

```
ssh2 username@ip-address -s xmlagent
```

ログインユーザー名、デバイスの IP アドレス、接続するサービスを入力します。xmlagent サービスは、デバイス ソフトウェアでは XML サーバと呼ばれます。

Note SSH コマンドの構文は、クライアント PC の SSH ソフトウェアによって異なることがあります。

XML サーバから hello メッセージを受信しなかった場合は、次の条件を確認してください。

- デバイスで SSH サーバがイネーブルになっています。
- XML サーバの *max-sessions* オプションは、デバイスへの SSH 接続の数をサポートするのに十分です。
- デバイス上の現用系 XML サーバ セッションの一部が使用されていません。

hello メッセージの送信

サーバーが他の要求を処理する前に、hello メッセージを使用して、サーバーに自分の機能をアドバタイズする必要があります。XML サーバーへの SSH セッションを開始すると、サーバーはすぐに hello メッセージで応答し、サーバーの機能をクライアントに通知します。このメッセージは、サーバーの機能をクライアントに通知します。XML サーバーは基本機能のみをサポートしているので、クライアントに対しても、基本機能のみのサポートを想定しています。

以下は、サーバーとクライアントからのサンプルの hello メッセージです。

Note すべての XML ドキュメントは、]]>]]> で終了して、SSH 経由の NETCONF で同期がサポートされるようにする必要があります。

サーバーからの hello メッセージ

```
<?xml version="1.0"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
  <capability>urn:ietf:params:xml:ns:netconf:base:1.0</capability>
  </capabilities>
  <session-id>25241</session-id>
</hello>]]>]]>
```

クライアントからの hello メッセージ

```
<?xml version="1.0"?>
<nc:hello xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <nc:capabilities>
  <nc:capability>urn:ietf:params:xml:ns:netconf:base:1.0</nc:capability>
  </nc:capabilities>
</nc:hello>]]>]]>
```

XML スキーマ定義（XSD）ファイルの取得

Procedure

Step 1 switch# feature bash shell

Step 2 switch# run bash

Step 3 bash-3.2$ cd /isan/etc/schema

Step 4 必要なスキーマを取得します。

XML ドキュメントを XML サーバに送信する

コマンド シェルで開いた SSH セッションを介して XML ドキュメントを XML サーバに送信するには、エディターから XML テキストをコピーして、SSH セッションに貼り付けます。通常、XML ドキュメントを XML サーバに送信するには自動化されたメソッドを使用しますが、このコピーと貼り付けによる方法を使えば、XML サーバへの SSH 接続を確認できます。

XML ドキュメントを XML サーバーに送信する場合のガイドラインは次のとおりです。

- コマンド シェル出力で Hello メッセージ テキストを検索して、SSH セッションを開始した直後に XML サーバーが Hello メッセージを送信したことを確認します。

- XML 要求を送信する前に、クライアントの Hello メッセージを送信します。XML サーバーは Hello 応答をすぐに送信するため、クライアント Hello メッセージを送信した後、追加の応答は送信されないことに注意してください。
- XML ドキュメントは常に文字シーケンス]]>]]> で終了します。

NETCONF XML インスタンスの作成

NETCONF XML インスタンスは、RPC タグおよび NETCONF 操作タグで XML デバイス要素を囲むことにより作成できます。XML デバイス要素は、使用可能な CLI コマンドを XML フォーマットで囲む機能ベースの XML スキーマ定義（XSD）ファイルで定義されます。

以下は、フレームワーク コンテキストの NETCONF XML リクエストで使用されるタグです。タグ ラインは次のレター コードでマークキングされています：

- X — XML 宣言
- R — RPC リクエスト タグ
- N — NETCONF 操作タグ
- D — デバイス タグ

NETCONF XML フレームワークのコンテキスト

```
X <?xml version="1.0"?>
R <nc:rpc message-id="1" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
R xmlns="http://www.cisco.com/nxos:1.0:nfcli”>
N <nc:get>
N <nc:filter type="subtree">
D <show>
D <xml>
D <server>
D <status/>
D </server>
D </xml>
D </show>
N </nc:filter>
N </nc:get>
R </nc:rpc>]]>]]>
```

Note 任意の XML エディタまたは XML 管理インターフェイス ツールを使用して、XML インスタンスを作成する必要があります。

RPC リクエスト タグ

すべての NETCONF XML インスタンスは、RPC リクエスト タグ <rpc> で開始する必要があります。は、<rpc>要素には、メッセージ ID（message-id）属性があります。message-id 属性は、<rpc-reply> 内に複製され、リクエストと応答を関連付けるために使用できます。<rpc> ノードもまた、次の XML 名前空間宣言も含まれています。

- NETCONF 名前空間宣言：`urn:ietf:params:xml:ns:netconf:base:1.0` 名前空間で定義されている <rpc> と NETCONF タグは、netconf.xsd スキーマ ファイルに存在します。
- デバイスの名前空間宣言：<rpc> と NETCONF タグによってカプセル化されたデバイス タグは、他の名前空間で定義されています。デバイスの名前空間は機能指向です。Cisco NX-OS 機能タグは、さまざまな名前空間で定義されています。RPC リクエスト タグ <rpc> は、NFCLI 機能の使用例です。これは、デバイスの名前空間が `xmlns=http://www.cisco.com/nxos:1.0:nfcli` であることを宣言しています。`nfcli.xsd` には、この名前空間の定義が含まれています。詳細については、XML スキーマ定義（XSD）ファイルの取得, on page 6を参照してください。

例

RPC リクエスト タグ <rpc>

```
<nc:rpc message-id="315" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns=http://www.cisco.com/nxos:1.0:nfcli">
...
</nc:rpc>]]>]]>
```

構成リクエスト

```
<?xml version="1.0"?>
<nc:rpc message-id="16" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns="http://www.cisco.com/nxos:1.0:if_manager">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
    <nc:config>
      <configure>
        <__XML__MODE__exec_configure>
          <interface>
            <ethernet>
              <interface>2/30</interface>
              <__XML__MODE_if-ethernet>
                <__XML__MODE_if-eth-base>
                  <description>
                    <desc_line>Marketing Network</desc_line>
                  </description>
                </__XML__MODE_if-eth-base>
              </__XML__MODE_if-ethernet>
            </ethernet>
          </interface>
        </__XML__MODE__exec_configure>
      </configure>
    </nc:config>
  </nc:edit-config>
</nc:rpc>]]>]]>
```

Note __XML__MODE タグは、NETCONF エージェントによって内部的に使用されます。一部のタグは、特定の __XML__MODE の子としてのみ存在します。スキーマファイルを調べると、XML で CLI コマンドを表すタグにつながる正しいモードタグを見つけることができます。

NETCONF 動作タグ

NETCONF は、次の構成動作を提供します。

Table 2: Cisco NX-OS の NETCONF 動作

NETCONF 動作	説明	例
close-session	XML サーバー セッションを閉じます。	NETCONF クロースセッションインスタンス, on page 18
commit	候補構成の現在の内容を実行構成として設定します。	NETCONF コミットインスタンス：候補構成機能, on page 23
confirmed-commit	指定された時間に構成を確定するためのパラメータを指定します。この操作の後の、確認タイムアウト期間内に確定操作が行われなかった場合、構成は確定確認操作の前の状態に戻ります。	NETCONF 確認済みコミットインスタンス, on page 23
copy-config	送信元構成データストアの内容をターゲットデータストアにコピーします。	NETCONF での構成インスタンスのコピー, on page 19
delete-config	操作がサポートされていません。	—
edit-config	デバイスの実行構成の機能を構成します。この操作は **configuration** コマンドに使用します。	NETCONF Edit Config インスタンス, on page 20 NETCONF Rollback-On-Error インスタンス, on page 24
get	デバイスから構成情報を受信します。この操作は **show** コマンドに使用します。データのソースは実行コンフィギュレーションです。	NETCONF XML インスタンスの作成, on page 7

NETCONF 動作	説明	例
get-config	構成の全体または一部を取得します。	NETCONF XML インスタンスの作成, on page 7
kill-session	指定した XML サーバー セッションを閉じます。自分のセッションを閉じることはできません。	NETCONF 強制終了セッションインスタンス, on page 18
ロック	クライアントがデバイスの構成システムをロックできるようにします。	NETCONF ロック インスタンス, on page 22
unlock	セッションが発行した構成ロックを解放します。	NETCONF ロック解除インスタンス, on page 23
検証	構成をデバイスに適用する前に、構成候補の構文エラーおよびセマンティクス エラーをチェックします。	NETCONF 検証機能インスタンス, on page 24

デバイスタグ

XML デバイス要素は、使用可能な CLI コマンドを XML フォーマットで表します。機能固有のスキーマ ファイルには、その特定の機能の CLI コマンドの XML タグが含まれています。「XML スキーマ定義（XSD）ファイルの取得, on page 6」を参照してください。

このスキーマを使用して、XML インスタンスを構築することができます。NETCONF インスタンスのビルドに使用された nfcli.xsd スキーマ ファイルの関連部分です。（NETCONF XML インスタンスの作成, on page 7 を参照）。

xml デバイス タグを表示します。

```
<xs:element name="show" type="show_type_Cmd_show_xml"/>
<xs:complexType name="show_type_Cmd_show_xml">
<xs:annotation>
<xs:documentation>to display xml agent information</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:choice maxOccurs="1">
<xs:element name="xml" minOccurs="1" type="xml_type_Cmd_show_xml"/>
<xs:element name="debug" minOccurs="1" type="debug_type_Cmd_show_debug"/>
</xs:choice>
</xs:sequence>
<xs:attribute name="xpath-filter" type="xs:string"/>
<xs:attribute name="uses-namespace" type="nxos:bool_true"/>
</xs:complexType>
```

サーバ ステータス デバイス タグ

```
<xs:complexType name="xml_type_Cmd_show_xml">
<xs:annotation>
```

```
<xs:documentation>xml agent</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:element name="server" minOccurs="1" type="server_type_Cmd_show_xml"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="server_type_Cmd_show_xml">
<xs:annotation>
<xs:documentation>xml agent server</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:choice maxOccurs="1">
<xs:element name="status" minOccurs="1" type="status_type_Cmd_show_xml"/>
<xs:element name="logging" minOccurs="1" type="logging_type_Cmd_show_logging_facility"/>
</xs:choice>
</xs:sequence>
</xs:complexType>
```

デバイスタグの応答

```
<xs:complexType name="status_type_Cmd_show_xml">
<xs:annotation>
<xs:documentation>display xml agent information</xs:documentation>
</xs:annotation>
<xs:sequence>
<xs:element name="__XML__OPT_Cmd_show_xml___readonly__" minOccurs="0">
<xs:complexType>
<xs:sequence>
<xs:group ref="og_Cmd_show_xml___readonly__" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:group name="og_Cmd_show_xml___readonly__">
<xs:sequence>
<xs:element name="__readonly__" minOccurs="1" type="__readonly___type_Cmd_show_xml"/>
</xs:sequence>
</xs:group>
<xs:complexType name="__readonly___type_Cmd_show_xml">
<xs:sequence>
<xs:group ref="bg_Cmd_show_xml_operational_status" maxOccurs="1"/>
<xs:group ref="bg_Cmd_show_xml_maximum_sessions_configured" maxOccurs="1"/>
<xs:group ref="og_Cmd_show_xml_TABLE_sessions" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
```

Note 「__XML__OPT_Cmd_show_xml___readonly__」タグはオプションです。このタグは応答を表します。応答の詳細については、RPC 応答タグ, on page 16 を参照してください。

<get> 操作を実行するために使用できるタグを見つけるための | XML オプションを使用できます。以下は | XML オプションの例です。この例は、このデバイスで操作を実行するための名前空間定義タグが `http://www.cisco.com/nxos:1.0:nfcli` であり、要求のビルドのためにnfcli.xsd ファイルを使用できることがわかります。

NETCONF 操作タグとデバイス タグを RPC タグで囲むことができます。</rpc> 終了タグの後に XML 終了文字シーケンスが続きます。

XML の例

```
Switch#> show xml server status | xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:nfcli">
<nf:data>
<show>
<xml>
<server>
<status>
<__XML__OPT_Cmd_show_xml___readonly__>
<__readonly__>
<operational_status>
<o_status>enabled</o_status>
</operational_status>
<maximum_sessions_configured>
<max_session>8</max_session>
</maximum_sessions_configured>
</__readonly__>
</__XML__OPT_Cmd_show_xml___readonly__>
</status>
</server>
</xml>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

拡張された NETCONF の操作

Cisco NX-OS は、<exec-command> という名前の<rpc> 操作をサポートします。この操作により、クライアントアプリケーションは CLI の **configuration** コマンドと **show** コマンドを送信し、それらのコマンドへの応答を XML タグとして受信できます。

以下は、インターフェイスの構成に使用されるタグの例です。タグラインは次のレターコードでマークキングされています：

- X — XML 宣言
- R — RPC リクエスト タグ
- EO — 拡張操作

次の表に、操作タグの詳細な説明を示します。

Table 3: 操作タグ

タグ	説明
<exec-command>	CLI コマンドの実行

タグ	説明
<cmd>	CLI コマンドが含まれています。コマンドは、**show** または **configuration** コマンドです。複数の構成コマンドは、セミコロン（;）を使用して区切ります。複数の **show** コマンドはサポートされていませんが、複数の構成コマンドを異なる <cmd> タグに含めて、同じリクエストの一部として送信することはできます。詳細については、*<exec-command>* を通して送信される構成 *CLI* コマンドの例を参照してください。

を介して送信される構成コマンドへの応答 <cmd> タグは次のとおりです。

- <nf:ok>：すべての **configuration** コマンドが正常に実行されました。
- <nf:rpc-error>：一部のコマンドが失敗しました。操作は最初のエラーで停止します。<nf:rpc-error> サブツリーは、失敗した構成について詳細情報を提供します。失敗したコマンドの前に実行された構成は、実行構成に適用されていることに注意してください。

<exec-command> を通して送信される構成 CLI コマンド

show コマンドは、次の例に示すように、それ自体の <exec-command> インスタンスに含めて送信する必要があります：

```
X <?xml version="1.0"?>
R <nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
EO <nxos:exec-command>
EO <nxos:cmd>conf t ; interface ethernet 2/1 </nxos:cmd>
EO <nxos:cmd>channel-group 2000 ; no shut; </nxos:cmd>
EO </nxos:exec-command>
R </nf:rpc>]]>]]>
```

<exec-command> を通して送信された CLI コマンドへの応答

送信操作に対する応答は次のとおりです：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
<nf:ok/>
</nf:rpc-reply>
]]>]]>
```

<exec-command> を通して送信された表示 CLI コマンド

次の例は、<exec-command> を通して送信される **show** CLI コマンドによってデータを取得する方法を示しています：

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nxos:exec-command>
<nxos:cmd>show interface brief</nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
```

<exec-command> を通して送信された表示 CLI コマンドへの応答

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0"
xmlns:mod="http://www.cisco.com/nxos:1.0:if_manager" message-id="110">
<nf:data>
<mod:show>
<mod:interface>
<mod:__XML__OPT_Cmd_show_interface_brief___readonly__>
<mod:__readonly__>
<mod:TABLE_interface>
<mod:ROW_interface>
<mod:interface>mgmt0</mod:interface>
<mod:state>up</mod:state>
<mod:ip_addr>192.0.2.20</mod:ip_addr>
<mod:speed>1000</mod:speed>
<mod:mtu>1500</mod:mtu>
</mod:ROW_interface>
<mod:ROW_interface>
<mod:interface>Ethernet2/1</mod:interface>
<mod:vlan>--</mod:vlan>
<mod:type>eth</mod:type>
<mod:portmode>routed</mod:portmode>
<mod:state>down</mod:state>
<mod:state_rsn_desc>Administratively down</mod:state_rsn_desc>
<mod:speed>auto</mod:speed>
<mod:ratemode>D</mod:ratemode>
</mod:ROW_interface>
</mod:TABLE_interface>
</mod:__readonly__>
</mod:__XML__OPT_Cmd_show_interface_brief___readonly__>
</mod:interface>
</mod:show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

機能不全の構成

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
<nxos:exec-command>
<nxos:cmd>configure terminal ; interface ethernet2/1 </nxos:cmd>
<nxos:cmd>ip address 192.0.2.2/24 </nxos:cmd>
<nxos:cmd>no channel-group 2000 ; no shut; </nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="3">
<nf:rpc-error>
<nf:error-type>application</nf:error-type>
<nf:error-tag>invalid-value</nf:error-tag>
<nf:error-severity>error</nf:error-severity>
<nf:error-message>Ethernet2/1: not part of port-channel 2000
</nf:error-message>
<nf:error-info>
<nf:bad-element>cmd</nf:bad-element>
</nf:error-info>
</nf:rpc-error>
</nf:rpc-reply>
]]>]]>
```

コマンドの実行により、インターフェイスの IP アドレスは設定されますが、管理状態は変更されません（**no shut** コマンドは実行されません）。管理状態が変更されない理由は、**no port-channel 2000** コマンドがエラーになったためです。

この <rpc-reply> は、ある **show** コマンドが、その **show** コマンドの XML 出力を含む <cmd> タグによって送信された結果です。

構成コマンドと表示コマンドを同じに組み合わせることはできません<exec-command> インスタンス。次は、同じインスタンスで **config** コマンドと **show** コマンドが組み合わされた例を示しています。

構成コマンドと表示コマンドの組み合わせ

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nxos:exec-command>
<nxos:cmd>conf t ; interface ethernet 2/1 ; ip address 1.1.1.4/24 ; show xml
server status </nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nf:rpc-error>
<nf:error-type>application</nf:error-type>
<nf:error-tag>invalid-value</nf:error-tag>
<nf:error-severity>error</nf:error-severity>
<nf:error-message>Error: cannot mix config and show in exec-command. Config cmds
before the show were executed.
Cmd:show xml server status</nf:error-message>
<nf:error-info>
<nf:bad-element>cmd</nf:bad-element>
</nf:error-info>
</nf:rpc-error>
</nf:rpc-reply>
]]>]]>
```

<exec-command> を通して送信された表示 CLI コマンド

```
<?xml version="1.0"?>
<nf:rpc xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
```

```
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nxos:exec-command>
<nxos:cmd>show xml server status ; show xml server status </nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nxos="http://www.cisco.com/nxos:1.0" message-id="110">
<nf:rpc-error>
<nf:error-type>application</nf:error-type>
<nf:error-tag>invalid-value</nf:error-tag>
<nf:error-severity>error</nf:error-severity>
<nf:error-message>Error: show cmds in exec-command shouldn't be followed by anything
</nf:error-message>
<nf:error-info>
<nf:bad-element><cmd></nf:bad-element>
</nf:error-info>
</nf:rpc-error>
</nf:rpc-reply>
]]>]]>
```

NETCONF 応答

クライアントによって送信されるすべての XML 要求に対して、XML サーバーは RPC 応答タグ <rpc-reply> で囲まれた XML 応答を送信します。

RPC 応答タグ

次の例は、RPC 応答タグ <rpc-reply> を示しています。

RPC 応答タグ<rpc-reply>

```
<nc:rpc-reply message-id="315" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns=http://www.cisco.com/nxos:1.0:nfcli">
<ok/>
</nc:rpc-reply>]]>]]>
```

RPC 応答エレメント

<ok>、<data>、そして <rpc-error> の要素は RPC 応答に表示される可能性があります。次の表は、<rpc-reply> タグに現れる可能性のある RPC 応答要素について説明しています。

Table 4: RPC 応答エレメント

要素	説明
<ok>	RPC 要求は正常に完了しました。この要素は、応答でデータが返されない場合に使用されます。
<data>	RPC 要求は正常に完了しました。RPC 要求に関連するデータは、<data> 要素を伴っています。

要素	説明
<rpc-error>	RPC 要求が失敗しました。エラー情報は、<rpc-error> 要素。

データ タグにカプセル化されたタグの解釈

<data> タグ内にカプセル化されたデバイス タグには、要求とそれに続く応答が含まれます。次の例に表示されているように、クライアント アプリケーションは、<readonly> タグ前のタグを全て安全に無視できます：

RPC 応答データ

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager">
<nf:data>
<show>
<interface>
<__XML__OPT_Cmd_show_interface_brief___readonly__>
<__readonly__>
<TABLE_interface>
<ROW_interface>
<interface>mgmt0</interface>
<state>up</state>
<ip_addr>xx.xx.xx.xx</ip_addr>
<speed>1000</speed>
<mtu>1500</mtu>
</ROW_interface>
<ROW_interface>
<interface>Ethernet2/1</interface>
<vlan>--</vlan>
<type>eth</type>
<portmode>routed</portmode>
<state>down</state>
<state_rsn_desc>Administratively down</state_rsn_desc>
<speed>auto</speed>
<ratemode>D</ratemode>
</ROW_interface>
</TABLE_interface>
</__readonly__>
</__XML__OPT_Cmd_show_interface_brief___readonly__>
</interface>
</show>
</nf:data>
</nf:rpc-reply>
]]>]]>
```

Note <__XML__OPT.*> と <__XML__BLK.*> はレスポンスに表示され、リクエストで使用されることもあります。これらのタグは NETCONF エージェントによって使用され、<__readonly__> タグの後の応答に存在します。これらは要求内で必須であり、スキーマ ファイルに従って追加した後、CLI コマンドを表す XML タグを続ける必要があります。

サンプル XML インスタンスに関する情報

XML インスタンスの例

このセクションでは、次の XML インスタンスの例を示します：

- NETCONF クロース セッション インスタンス, on page 18
- NETCONF 強制終了セッション インスタンス, on page 18
- NETCONF での構成インスタンスのコピー, on page 19
- NETCONF Edit Config インスタンス, on page 20
- NETCONF の Get 構成インスタンス, on page 21
- NETCONF ロック インスタンス, on page 22
- NETCONF ロック解除インスタンス, on page 23
- NETCONF コミット インスタンス：候補構成機能, on page 23
- NETCONF 確認済みコミット インスタンス, on page 23
- NETCONF Rollback-On-Error インスタンス, on page 24
- NETCONF 検証機能インスタンス, on page 24

NETCONF クロース セッション インスタンス

次の例は、close-session 要求とそれに続く close-session 応答を表示しています。

セッション要求の終了

```
<?xml version="1.0"?>
<nc:rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0">
<nc:close-session/>
</nc:rpc>]]>]]>
```

クローズセッション応答

```
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0" message-id="101">
<nc:ok/>
</nc:rpc-reply>]]>]]>
```

NETCONF 強制終了セッション インスタンス

次の例は、セッションの強制終了要求と、セッションの強制終了の応答を示しています。

セッションの強制終了要求

```
<nc:rpc message-id="101" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0">
<nc:kill-session>
<nc:session-id>25241</nc:session-id>
</nc:kill-session>
</nc:rpc>]]>]]>
```

セッションの終了応答

```
<?xml version="1.0"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0" message-id="101">
<nc:ok/>
</nc:rpc-reply>]]>]]>
```

NETCONF での構成インスタンスのコピー

Note <startup/>は、ソースまたはターゲットのデータストアとしてサポートされていません。コマンドを入力するなどしてコピー操作を実行するには、 にフォールバックする必要があります。<exec-command>メソッドを使用します。**startup-configcopy running-config startup-config**

次に、設定のコピー要求と、それに続く設定のコピー応答の例を示します。

構成コピーの要求

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<copy-config>
<target>
<running/>
</target>
<source>
<url>https://user@example.com:passphrase/cfg/new.txt</url>
</source>
</copy-config>
</rpc>
```

設定応答のコピー

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

NETCONF Edit Config インスタンス

Note 候補データストアを使用した XML edit-config は、バージョン 1.0 の XML 要求ではサポートされていません。ツールで xml を使用して生成できる新しいバージョンでのみサポートされます。

次の例は、NETCONF edit config の使用方法を示しています。

設定リクエストの編集

```
<?xml version="1.0"?>
<nc:rpc message-id="16" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager">
<nc:edit-config>
<nc:target>
<nc:running/>
</nc:target>
<nc:config>
<configure>
<__XML__MODE__exec_configure>
<interface>
<ethernet>
<interface>2/30</interface>
<__XML__MODE_if-ethernet>
<__XML__MODE_if-eth-base>
<description>
<desc_line>Marketing Network</desc_line>
</description>
</__XML__MODE_if-eth-base>
</__XML__MODE_if-ethernet>
</ethernet>
</interface>
</__XML__MODE__exec_configure>
</configure>
</nc:config>
</nc:edit-config>
</nc:rpc>]]>]]>
```

設定応答の編集

```
<?xml version="1.0"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://www.cisco.com/nxos:1.0:if_manager" message-id="16">
<nc:ok/>
</nc:rpc-reply>]]>]]>
```

edit-config の operation 属性は、指定された操作が実行される構成のポイントを指定します。操作属性が指定されていない場合、構成は既存の構成データ ストアにマージされます。操作属性には、次の値を指定できます。

- create
- merge
- delete

Edit Config：削除操作要求

次の例は、実行構成からインターフェイス Ethernet 0/0 の構成を削除する方法を示しています。

```
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
<target>
<running/>
</target>
<default-operation>none</default-operation>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<top xmlns="http://example.com/schema/1.2/config">
<interface xc:operation="delete">
<name>Ethernet0/0</name>
</interface>
</top>
</config>
</edit-config>
</rpc>]]>]]>
```

Edit Config 応答：削除操作

次の例は、実行構成のインターフェイス Ethernet 0/0 の構成を編集する方法を示しています。

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>]]>]]>
```

NETCONF の Get 構成インスタンス

次の例は、NETCONF の get 構成 の使用方法を示しています：

サブツリー全体を取得するための Get 構成要求

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
<source>
<running/>
</source>
<filter type="subtree">
<top xmlns="http://example.com/schema/1.2/config">
<users/>
</top>
</filter>
</get-config>
</rpc>]]>]]>
```

クエリの結果を含む Get 構成応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
<top xmlns="http://example.com/schema/1.2/config">
<users>
<user>
<name>root</name>
<type>superuser</type>
<full-name>Charlie Root</full-name>
<company-info>
<dept>1</dept>
<id>1</id>
</company-info>
</user>
<!-- additional <user> elements appear here... -->
</users>
</top>
</data>
</rpc-reply>]]>]]>
```

NETCONF ロック インスタンス

次の例は、ロック要求、成功の応答、および失敗した試行への応答を示しています。

ロック要求

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<lock>
<target>
<running/>
</target>
</lock>
</rpc>]]>]]>
```

ロックの取得に成功した場合の応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/> <!-- lock succeeded -->
</rpc-reply>]]>]]>
```

ロックの取得に失敗した場合の応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<rpc-error> <!-- lock failed -->
<error-type>protocol</error-type>
<error-tag>lock-denied</error-tag>
<error-severity>error</error-severity>
<error-message>
Lock failed, lock is already held
</error-message>
<error-info>
```

```
<session-id>454</session-id>
<!-- lock is held by NETCONF session 454 -->
</error-info>
</rpc-error>
</rpc-reply>]]>]]>
```

NETCONF ロック解除インスタンス

次の例は、NETCONF ロック解除の使用方法を示しています。

ロック解除要求

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<unlock>
<target>
<running/>
</target>
</unlock>
</rpc>
```

ロック解除要求への応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>
```

NETCONF コミット インスタンス：候補構成機能

次の例は、コミット操作とコミット返信を示しています。

操作をコミット

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit/>
</rpc>
```

返信をコミット

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>
```

NETCONF 確認済みコミット インスタンス

次の例は、確認済みのコミット操作と確認済みのコミット返答を示しています。

確認されたコミット リクエスト

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit>
<confirmed/>
<confirm-timeout>120</confirm-timeout>
</commit>
</rpc>]]>]]>
```

確認されたコミット 応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>]]>]]>
```

NETCONF Rollback-On-Error インスタンス

次の例は、エラー時のロールバックと、この要求への応答を構成する方法を示しています。

Rollback-On-Error の機能

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
<target>
<running/>
</target>
<error-option>rollback-on-error</error-option>
<config>
<top xmlns="http://example.com/schema/1.2/config">
<interface>
<name>Ethernet0/0</name>
<mtu>100000</mtu>
</interface>
</top>
</config>
</edit-config>
</rpc>]]>]]>
```

Rollback-On-Error への応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>]]>]]>
```

NETCONF 検証機能インスタンス

次の例は、NETCONF 検証機能の使用方法を示しています。文字列 urn:ietf:params:netconf:capability:rollback-on-error:1.0 は、NETCONF 検証機能を指定します。

検証要求

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

検証要求への応答

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ok/>
</rpc-reply>]]>]]>
```

その他の参考資料

ここでは、XML 管理インターフェイスの実装に関する追加情報について説明します。

RFC

RFC	タイトル
RFC 4741	NETCONF Configuration Protocol
RFC 4742	セキュアシェル（SSH）経由の NETCONF 構成プロトコルの使用

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。