XML 管理インターフェイス

- XML 管理インターフェイス，1 ページ

XML 管理インターフェイス

この章では、XML 管理インターフェイスを使用したデバイスの設定方法について説明します。

XML 管理インターフェイスの機能履歴

次の表に、この機能のリリースの履歴を示します。

表 *1*：*XML* 管理インターフェイスの機能履歴

機能名	リリース	機能情報
XML 管理インターフェイス	7.3(0)D1(1)	次の NetConf の拡張機能が追加されました。 • get-config • copy-config • validate • commit • lock • unlock • rollback on error、stop on error、continue on error、default operations、candidate config などのアクションをサポートするための edit-config の拡張

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を使用してデバイスを設定できます。このインターフェイスは、XML ベースのネットワーク設定プロトコル（NETCONF）を使用します。このプロトコルにより、XML 管理ツールまたはプログラムを使用したデバイスの管理とインターフェイスを介した通信が可能になります。Cisco NX-OS に NETCONF を実装するには、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、XML スキーマ（XSD）を使用して実装されます。この XML スキーマにより、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設定要素を組み込むことができます。ユーザは、RPC メッセージ内から、デバイスで実行するコマンドのタイプに一致する NETCONF 動作のいずれかを選択します。NETCONF を使用して、CLI コマンドのすべてのセットをデバイスに設定できます。NETCONF の使用については、NETCONF XML インスタンスの作成, （7 ページ）および RFC 4741 を参照してください。

SSH を通じた NETCONF の使用についての詳細は、RFC 4742 を参照してください。

この項では、次のトピックについて取り上げます。

NETCONF レイヤ

NETCONF レイヤを次に示します。

表 2 : *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>
動作	<get-config>、<edit-config>
コンテンツ	show コマンドまたは設定コマンド

次に、4 つの NETCONF レイヤの説明を示します。

- SSH トランスポート プロトコル — クライアントとサーバ間のセキュアな暗号化接続を提供します。
- RPC タグ — 要求者からの設定コマンドおよび XML サーバからの対応応答を示します。
- NETCONF 動作タグ — 設定コマンドのタイプを示します。
- コンテンツ — 設定する機能の 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 を通じた NETCONF をサポートする SSH サービス（xmlagent）を提供します。

（注） xmlagent サービスは、Cisco NX-OS ソフトウェアにおいて XML サーバと呼ばれます。

SSH を通じた NETCONF は、クライアントと XML サーバ間での hello メッセージの交換によって開始されます。最初の交換後、クライアントが XML 要求を送信し、サーバがこれに対して XML 応答を送信します。クライアントおよびサーバは、要求および応答の終端に文字シーケンス > を使用します。この文字シーケンスはXMLでは有効でないため、クライアントおよびサーバはメッセージの終端部と解釈することができ、通信の同期が維持されます。

使用可能な XML 設定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, （7 ページ）で説明します。

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 管理インターフェイスには、ライセンスは不要です。ライセンス パッケージに含まれていない機能は Cisco NX-OS 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には次の前提条件があります。

- SSHv2 がクライアント PC にインストールされていること。
- SSH を通じた NETCONF をサポートする XML 管理ツールがクライアント PC にインストールされていること。
- デバイスの XML サーバで適切なオプションが設定されていること。

XML 管理インターフェイスの使用

ここでは、XML 管理インターフェイスの手動設定と使用方法について説明します。XML 管理インターフェイスは、デバイスのデフォルト設定で使用します。

CLI による SSH および XML サーバ オプションの設定

デフォルトでは、SSH サーバがデバイスで有効化されています。SSH が無効になっている場合は、クライアント PC で SSH セッションが開始される前に、SSH を有効化する必要があります。

XML サーバ オプションの設定により、同時セッション数およびアクティブ セッションのタイムアウトを制御できます。また、XML ドキュメントの検証を有効化したり、XML セッションを終了することもできます。

（注） XML サーバのタイムアウトは、アクティブ セッションのみに適用されます。

SSH の設定に関する詳細については、ご使用のプラットフォームに対応した 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に対応した Cisco NX-OS システム管理構成ガイドを参照してください。

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configureterminal**	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	show **xmlserverstatus**	（任意）XML サーバ設定およびアクティブな XML サーバ セッションに関する情報を表示します。セッション番号は、コマンド出力から確認できます。
ステップ 3	**xmlservervalidateall**	指定したサーバセッションに対応する XML ドキュメントが検証されます。
ステップ 4	**xmlserverterminate***session*	指定した XML サーバ セッションを終了します。
ステップ 5	**nofeaturessh**	（任意）キーを生成できるように、SSH サーバを無効化します。 キーの生成については、次を参照してください。 SSH および XML サーバ オプションの設定
ステップ 6	**featuressh**	SSH サーバをイネーブルにします。デフォルトではイネーブルになっています。
ステップ 7	**showsshserver**	（任意）SSH サーバのステータスを表示します。
ステップ 8	**xmlservermax-session***sessions*	許容される XML サーバのセッション数を設定します。 デフォルト値は 8 です。有効な範囲は 1 ～ 8 です。
ステップ 9	**xmlservertimeout***seconds*	XML サーバ 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。 デフォルトは 1200 秒です。範囲は 1 ～ 1200 です。

	コマンドまたはアクション	目的
ステップ 10	show **xmlserverstatus**	（任意）XML サーバ設定およびアクティブな XML サーバ セッションに関する情報を表示します。
ステップ 11	copy running-config startup-config	（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。

次の例は、CLI を通じて SSH および XML サーバ オプションを設定する方法を示しています

```
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 2400
switch(config)# copy running-config startup-config
```

SSH 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、次のようなコマンドを使用できます。

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

ログイン ユーザ名、デバイスの IP アドレス、接続先のサービスを 入力します。xmlagent サービスは、デバイス ソフトウェアにおいて XML サーバと呼ばれます。

（注） SSH コマンドの構文は、クライアント PC の SSH ソフトウェアと異なる場合があります。

XML サーバから hello メッセージを受信しない場合は、以下の条件を確認してください。

- SSH サーバがデバイス上で有効化されていること。
- XML サーバの最大セッション オプションが、デバイスへの SSH 接続数に対応できるように適切に設定されていること。
- デバイス上のアクティブ XML サーバ セッションのすべてが使用中でないこと。

Hello メッセージの送信

XML サーバへの SSH セッションを開始すると、サーバはサーバの処理能力をクライアントに通知する hello メッセージが即座に応答されます。サーバが他の要求を処理する前に、クライアントの処理能力を hello メッセージでサーバにアドバタイズする必要があります。XML サーバでは、基本的な処理能力のみをサポートし、クライアントの基本的な処理能力に対するサポートのみを想定します。

次に、サーバおよびクライアントからの hello メッセージの例を示します。

（注） すべての XML ドキュメントは、]]>]]> で終了して、SSH 経由の NETCONF の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>]]>]]>
```

XSD ファイルの取得

ステップ 1 ブラウザで次の URL のシスコ ソフトウェア ダウンロード サイトに移動します。
http://software.cisco.com/download/navigator.html

[Download Software] ページが表示されます。

ステップ 2 [Select a Product] リストから、[Switches] > [DataCenterSwitches] > **[platform]** > [model] を選択します。

ステップ 3 登録シスコ ユーザとしてログインしていない場合は、すぐにログインするように求められます。

ステップ 4 [Select a Software Type] リストから、[NX-OSXMLSchemaDefinition] を選択します。

ステップ 5 目的のリリースを選択し、[Download] をクリックします。

ステップ 6 要求された場合には、強力な暗号化ソフトウェアイメージをダウンロードする資格申請の手順に従ってください。
[Cisco End User License Agreement] が開きます。

ステップ 7 [Agree] をクリックし、手順に従ってファイルを PC にダウンロードします。

XML サーバへの XML ドキュメントの送信

コマンド シェルで開いた SSH セッションを通じて XML ドキュメントを XML サーバに送信するには、XML テキストをエディタからコピーし、それを SSH セッションに貼り付けるという方法

があります。通常は自動化メソッドを使用して XML ドキュメントを XML サーバに送信しますが、この方法では XML サーバへの SSH 接続の確認ができます。

この方法を使用する場合は、以下のガイドラインに従ってください。

- シェル コマンドの出力で hello メッセージを検索して、SSH セッションを開始した直後に XML サーバから Hello メッセージが送信され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 hello メッセージを送信します。XML サーバから hello 応答が即座に送信されるため、クライアント hello メッセージを送信した後に追加の応答は送信されません。
- XML 文書は、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

RPC タグおよび NETCONF 動作タグ内に XML デバイス要素を組み込むことにより、NETCONF XML インスタンスを作成できます。XML デバイス要素は、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。このファイルには、使用可能な CLI コマンドが XML 形式で組み込まれます。

次に、フレームワーク コンテキストの NETCONF XML 要求で使用されるタグを示します。タグ行は、次の文字コードでマークされています。

- X — XML 宣言
- R — RPC 要求タグ
- N — NETCONF 動作タグ
- D — デバイス タグ

表 *3*：*NETCONF XML* フレームワーク 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>]]>]]>
```

（注） 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 要求タグ rpc

すべての NETCONF XML インスタンスの最初に RPC 要求タグ <rpc> を付ける必要があります。*RPC* 要求のタグ *<rpc>* の例は、<rpc> 要素と必要な **message-id** 属性を示しています。message-id 属性は、<rpc-reply> で複製され、要求と応答を関連付けるために使用できます。また、<rpc> ノードには次の XML 名前空間宣言が含まれています。

- NETCONF 名前空間宣言 — <rpc> および "urn:ietf:params:xml:ns:netconf:base:1.0" 名前空間で定義された NETCONF タグは、netconf.xsd スキーマ ファイルに存在します。
- デバイス名前空間宣言 — <rpc> タグと NETCONF タグでカプセル化されたデバイス タグは、別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 機能タグは別の名前空間で定義されています。*RPC* 要求のタグ *<rpc>* は、nfcli 機能を使用した例です。デバイスの名前空間が "xmlns=http://www.cisco.com/nxos:1.0:nfcli" と宣言されています。nfcli.xsd には、この名前空間定義が含まれています。詳細については、「*XSD* ファイルの取得」を参照してください。

表 *4*： *RPC* 要求タグ *<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>]]>]]>
```

設定要求の例を次に示します。

表 *5*： 設定要求

```
<?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>]]>]]>
```

__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 タグ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 の CLI コマンドを表すタグにつながる 適切なモード タグが見つかります。

NETCONF 動作タグ

NETCONF では、次の設定動作を提供します。

表 6：*Cisco NX-OS* での *NETCONF* 動作

NETCONF 動作	説明	例
close-session	現在の XML サーバセッションを閉じます。	NETCONF close-session インスタンス, （18 ページ）
commit	実行コンフィギュレーションを、候補コンフィギュレーションの現在の内容に設定します。	NETCONF commit インスタンス - 候補コンフィギュレーション機能, （23 ページ）
confirmed-commit	指定された期間、設定を確定するパラメータを提供します。この動作の後、confirmed-commit 期間内にコミット動作が実行されない場合、設定は confirmed-commit 動作前の状態に戻ります。	NETCONF confirmed-commit インスタンス , （24 ページ）
copy-config	コピー元の設定データストアの内容を対象のデータストアにコピーします。	NETCONF copy-config インスタンス, （19 ページ）
delete-config	この動作はサポートされていません。	—
edit-config	デバイスの実行コンフィギュレーションの機能を設定します。この動作は、コンフィギュレーション コマンドに使用します。 リリース 7.3(0)D1(1) 以降、create、delete、merge、rollback-on-error、continue-on-error、stop-on-error のアクションに対するサポートが追加されました。	NETCONF edit-config インスタンス, （20 ページ）NETCONF rollback-on-error インスタンス , （24 ページ）

NETCONF 動作	説明	例
get	デバイスから設定情報を受信します。この動作は、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, （7 ページ）
get-config	設定のすべてまたは一部を取得します。	NETCONF get-config インスタンス, （21 ページ）
kill-session	指定された XML サーバ セッションを閉じます。自身のセッションを閉じることはできません。NETCONF close-session 動作を参照してください。	NETCONF kill-session インスタンス, （19 ページ）
lock	クライアントがデバイスの設定システムをロックすることができます。	NETCONF lock インスタンス, （22 ページ）
unlock	以前のセッションで発行された設定ロックを解除します。	NETCONF unlock インスタンス, （23 ページ）
validate	デバイスに設定を適用する前に、候補コンフィギュレーションに構文エラーやセマンティック エラーがないかを確認します。	NETCONF 検証機能インスタンス , （25 ページ）

デバイス タグ

XML デバイス要素は、XML 形式で使用可能な CLI コマンドを示します。機能固有のスキーマ ファイルには、その特定機能の CLI コマンドに対応する XML タグが含まれます。XSD ファイルの取得, （6 ページ）の項を参照してください。

このスキーマを使用することにより、XML インスタンスの構築が可能になります。たとえば、NETCONF XML インスタンスの作成, （7 ページ）の構築に使用された nfcli.xsd スキーマ ファイルの関連部分を次の例に示します。

表 *7*： ***show xml*** デバイス タグ

```
<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>
```

表 *8*： ***server status*** 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>
```

次の例に、デバイス タグの応答を示します。

表 *9*：デバイスのタグの応答

```
<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>
```

（注）「__XML__OPT_Cmd_show_xml___readonly__」はオプションです。このタグは応答を表しています。応答の詳細については、RPC 応答タグ，（17 ページ）の項を参照してください。

| XML オプションを使用することにより、<get> の実行に使用する必要のあるタグを検索できます。次に、| XML オプションの例を示します。

表 *10*：*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>
]]>]]>
```

この応答から、このコンポーネント上で動作を実行するための 空間定義タグは http://www.cisco.com/nxos:1.0:nfcli であり、nfcli.xsd ファイルが この機能の要求を構成するために使用できることがわかります。

NETCONF 動作タグおよびデバイス タグを、RPC タグ内に 組み込むことができます。</rpc> 終了タグの後は、XML 終端文字 シーケンスです。

NETCONF の拡張動作

Cisco NX-OS は、<exec-command> という名前の <rpc> 動作をサポートします。クライアント アプリケーションでは、この動作を使用することにより、CLI設定を送信してコマンドを表示したり、これらのコマンドの応答を XML タグとして受信することができます。

以下は、インターフェイスの設定に使用されるタグの例です。タグ行は、次の文字コードでマークされています。

- X — XML 宣言
- R — RPC 要求タグ
- EO — 拡張動作

表 *11*： *<exec-command>* によって送信される設定 *CLI* コマンド

```
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>]]>]]>
```

以下は、動作に対する応答です。

表 *12*： *<exec-command>* によって送信された *CLI* コマンドに対する応答

```
<?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" message-id="3">
<nf:ok/>
</nf:rpc-reply>
]]>]]>
```

次の例は、<exec-command> で送信された show CLI コマンドを 使用してデータを取得する方法を示しています。

表 *13*： ***<exec-command>*** によって送信される ***show 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 interface brief</nxos:cmd>
</nxos:exec-command>
</nf:rpc>]]>]]>
```

以下は、動作に対する応答です。

表 *14*： ***<exec-command>*** によって送信された ***show CLI*** コマンドに対する応答

```
<?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>172.23.152.20</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>
]]>]]>
```

次の表に、動作タグの詳細な説明を 示します。

表 *15*：タグ

タグ	説明
<exec-command>	CLI コマンドを実行します。

タグ	説明
<cmd>	CLI コマンドを含みます。show コマンドまたは設定コマンドを含むことができます。複数の設定コマンドを含む場合は、セミコロン「;」で区切ます。複数の show コマンドはサポートされていません。複数の設定コマンドは、複数の <cmd> タグで同じ要求の一部として送信できます。詳細については、「*<exec-command>* によって送信される設定 *CLI* コマンド」の例を参照してください。

<cmd> タグで送信された設定コマンドに対する応答は以下のとおりです。

- <nf:ok>：すべての設定コマンドが正常に実行されています。
- <nf:rpc-error>：一部のコマンドでエラーが発生しました。動作は最初のエラーの発生時に停止し、<nf:rpc-error> サブツリーに設定のエラー内容の詳細が示されます。失敗したコマンドより前に実行された設定のすべては、実行中の設定にすでに適用されている点に注意してください。

次に、失敗した設定の例を示します。

表 *16*： 設定の失敗

```
<?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 1.1.1.2/24 </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>
]]>]]>
```

コマンドを実行した結果、インターフェイスの IP アドレスが設定 されましたが、no port-channel 2000 コマンドの結果がエラーであるため、管理状態が変更されていません（no shut コマンドが実行されていない）。

<cmd> タグで送信された show コマンドの結果として <rpc-reply> に show コマンドの XML 出力が含まれます。

同一の <exec-command> インスタンス上で設定コマンドと show コマンドを組み合わせることはできません。同一のインスタンス内で show コマンドと 設定コマンドを組み合わせた 例を次に示します。

表 *17*： 設定コマンドと *show* 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>
]]>]]>
```

show コマンドは、次の例に示すように、個別の <exec-command> インスタンスで送信される必要があります。

表 *18*： *<exec-command>* によって送信される *show 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>&lt;cmd&gt;</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たすべての XML 要求に対し、XML サーバは RPC 応答タグ <rpc-reply> で囲んだ XML 応答を送信します。

ここでは、次の内容について説明します。

RPC 応答タグ

次の例は、RPC 応答タグ <rpc-reply> を示しています。

表 *19*：*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>]]>]]>
```

要素 <ok>、<data>、<rpc-error> が RPC 応答に 表示されることがあります。次の表に、<rpc-reply> タグに表示される可能性のある RPC 応答要素について説明します。

表 *20*：*RPC* 応答要素

要素	説明
<ok>	RPC 要求は正常に完了しています。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ています。RPC 要求に関連付けられたデータは<data> 要素で囲まれています。
<rpc-error>	RPC 要求が失敗しました。エラー情報は <rpc-error> 要素で囲まれています。

データ タグにカプセル化されたタグの解釈

<data> タグによってカプセル化されたデバイス タグには、要求が含まれ、その後に応答が続きます。クライアント アプリケーションは、<readonly> タグより前のすべてのタグを安全に無視することができます。次に、例を示します。

表 *21*： ***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>
]]>]]>
```

<__XML__OPT.*> および <__XML__BLK.*> は、 応答の中で使用され、要求内でも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、要求において必須であり、 CLI コマンドを表す XML タグに到達するためにスキーマ ファイルに従って 追加される必要があります。

XML インスタンスの例

ここでは、次の XML インスタンスの例を示します。

NETCONF close-session インスタンス

次の例は、close-session 要求とその後の close-session 応答を示しています。

表 *22*： ***close-session*** 要求

```
<?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>]]>]]>
```

表 *23*： ***close-session*** 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 kill-session インスタンス

次の例は、kill-session 要求とその後の kill-session 応答を示しています。

表 *24*： ***kill-session*** 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>]]>]]>
```

表 *25*： ***kill-session*** 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 copy-config インスタンス

次の例は、copy-config 要求とその後の copy-config 応答を示しています。

表 *26*： ***copy-config*** 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>
```

表 *27*： ***copy-config*** 応答

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

NETCONF edit-config インスタンス

次の例は、NETCONF edit-config の使用を示しています。

表 *28*：*edit-config* 要求

```
<?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>]]>]]>
```

表 *29*：*edit-config* 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:if_manager" message-id="16">
<nc:ok/>
</nc:rpc-reply>]]>]]>
```

edit-config の動作属性は、 指定された動作が実行される設定ポイントを 特定します。動作属性が指定されていない場合、 その設定は既存の設定データ ストアに マージされます。動作属性には次の値を設定できます。

- create
- merge
- delete

次の例は、実行コンフィギュレーションからイーサネット 0/0 インターフェイスの設定を削除する方法を示しています。

表 *30*： ***edit-config***：動作要求の削除

```
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>]]>]]>
```

表 *31*： ***edit-config*** に対する応答：動作の削除

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

NETCONF get-config インスタンス

次の例は、NETCONF get-config の使用を示しています。

表 *32*：サブツリー全体を取得する ***get-config*** 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>]]>]]>
```

表 *33*： ***get-config*** 応答とクエリの結果

```
<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 lock インスタンス

次の例は、NETCONF の lock 動作の使用を示しています。

次の例は、lock 要求、成功応答、失敗した試行に対する応答を示しています。

表 *34*： ***lock*** 要求

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

表 *35*： 成功した ***lock*** 取得に対する応答

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

表 *36*： 失敗した *lock* 試行に対する応答

```
<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 unlock インスタンス

次の例は、NETCONF の unlock 動作の使用を示しています。

表 *37*： *unlock* 要求

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

表 *38*： *unlock* 要求に対する応答

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

NETCONF commit インスタンス - 候補コンフィギュレーション機能

次の例は、commit 動作と commit 応答を示しています。

表 *39*： *commit* 動作

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

表 *40*： ***commit*** 応答

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

NETCONF confirmed-commit インスタンス

次の例は、confirmed-commit 動作とその後の confirmed-commit 応答を示しています。

表 *41*： ***confirmed-commit*** 要求

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

表 *42*： ***confirmed-commit*** 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、文字列 urn:ietf:params:netconf:capability:rollback-on-error:1.0 で指定された NETCONF のエラー時ロールバック機能の使用を示しています。

次の例は、エラー時ロールバックの設定方法と、この要求に対する応答を示しています。

表 *43*： ***rollback-on-error*** 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>]]>]]>
```

表 *44*： ***rollback-on-error*** 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、文字列 **urn:ietf:params:netconf:capability:validate:1.0** で指定された NETCONF 検証機能の使用を示しています。

表 *45*：検証要求

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

表 *46*：検証要求に対する応答

```
<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を提供します。

標準

標準	タイトル
この機能がサポートする新しい規格または変更された規格はありません。既存の規格のサポートは、この機能によって変更されていません。	—

RFC

RFC	タイトル
RFC 4741	『NETCONF Configuration Protocol』
RFC 4742	『Using the NETCONF Configuration Protocol over Secure Shell (SSH)』