NETCONF プロトコルを使用したデータ モデルによるネットワーク運用の定義

XR デバイスには、サポート対象のデータ モデルを定義する YANG ファイルが付属しています。NETCONF や gRPC などの管理プロトコルを使用して、デバイスに対してサポートされているモデルのリストをプログラムで検索し、モデル ファイルを取得することができます。

ネットワーク設定プロトコル（NETCONF）は、ネットワーク デバイスと通信する標準的なトランスポート プロトコルです。NETCONF は、設定データを編集し、ネットワーク デバイスから運用データを取得するためのメカニズムを提供します。設定データは、インターフェイス、ルーティング プロトコル、およびその他のネットワーク機能をプロビジョニングする方法を表します。運用データは、インターフェイスの統計情報、メモリ使用率、エラーなどを表します。

NETCONF は設定データとプロトコル メッセージに Extensible Markup Language（XML）ベースのデータ符号化を使用します。シンプルな RPC（リモート プロシージャ コール）ベースのメカニズムを使用してクライアントとサーバ間の通信を促進します。クライアントはネットワーク マネージャの一部として実行されているスクリプトやアプリケーションです。サーバは、ルータなどのネットワーク デバイスです。NETCONF はデバイスとの通信方法を定義しますが、クライアントとサーバ間で交換するデータを処理することはありません。

NETCONF を有効にするには、**ssh server capability netconf-xml** コマンドを使用してポート 22 の XML サブシステムにアクセスします。

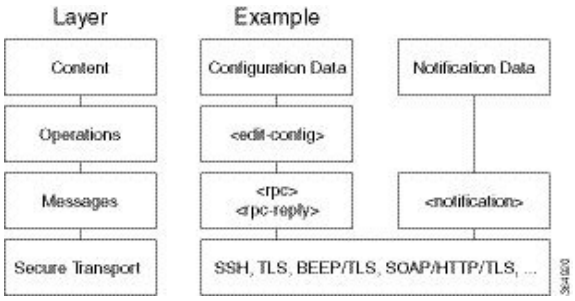
NETCONF セッション

NETCONF セッションは、ネットワーク コンフィギュレーション アプリケーション（クライアント）とネットワーク デバイス（ルータ）の間の論理接続です。設定の属性は承認されたセッション時に変更でき、その影響はすべてのセッションに反映されます。NETCONF はコネクション型で、基盤となる転送に SSH を使用しています。NETCONF セッションは、機能が公開されている場合は「hello」メッセージで確立され、*close* メッセージまたは *kill* メッセージを使用して終了します。

NETCONF レイヤ

NETCONF プロトコルは 4 つのレイヤに分割できます。

図 *1 : NETCONF* レイヤ



- **コンテンツ レイヤ**：設定および通知データが含まれます
- **動作レイヤ**：XML で符号化されたパラメータによる RPC メソッドとして起動される一連の基本プロトコル動作を定義します
- **メッセージ レイヤ**：RPC と通知を符号化するためのシンプルな、トランスポート非依存のフレーミング メカニズムを提供します
- **セキュア トランスポート レイヤ**：クライアントとサーバ間の通信パスを提供します

NETCONF の詳細については、RFC 6241 を参照してください。

ここでは、ルータのローカル時刻を設定する使用例を用いて、CLI と比較し、データ モデルが高速プログラム設定でどのように役立つかについて説明します。

- NETCONF の操作 （2 ページ）
- NETCONF セッションでのデータ モデルを使用したルータ クロックの設定 （6 ページ）

NETCONF の操作

NETCONF は 1 つ以上のコンフィギュレーション データストアの存在を定義し、それらでの設定操作を可能にします。設定データストアは、初期のデフォルト状態から必要な動作状態にデバイスを移行するために必要な一連の完全なコンフィギュレーション データです。コンフィギュレーション データストアには状態データやエグゼクティブ コマンドは含まれません。

基本プロトコルには、次の NETCONF 操作が含まれています。

```
|  +--get-config
|  +--edit-Config
|     +--merge
|     +--replace
|     +--create
|     +--delete
|     +--remove
|     +--default-operations
|        +--merge
|        +--replace
|        +--none
|  +--get
|  +--lock
|  +--unLock
```

```
|   +--close-session
|   +--kill-session
```

これらの NETCONF 操作については、次の表で説明します。

NETCONF 動作	説明	例
<get-config>	指定した設定の全部または一部を名前付きのデータストアから取得します	フィルタ オプションを使用して実行中の設定から特定のインターフェイス設定の詳細を取得します `<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><get-config><source><running/></source><filter><interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg"\><interface-configuration><active>act</active><interface-name>TenGigE0/0/0/2</interface-name></interface-configuration></interface-configurations></filter></get-config></rpc>`
<get>	実行中の設定およびデバイスの状態情報を取得します	すべての ACL 設定およびデバイスの状態情報を取得します `Request:<get><filter><ipv4-acl-and-prefix-list xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-acl-oper"/></filter></get>`

NETCONF 動作	説明	例
<edit-config>	指定した設定の全部または一部を指定したターゲット設定にロードします	**Merge** 操作で ACL 設定を設定します `<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <edit-config> <target><candidate/></target> <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"> <ipv4-acl-and-prefix-list xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-acl-cfg" xc:operation=”merge”> <accesses> <access> <access-list-name>aclv4-1</access-list-name> <access-list-entries> <access-list-entry> <sequence-number>10</sequence-number> <remark>GUEST</remark> </access-list-entry> <access-list-entry> <sequence-number>20</sequence-number> <grant>permit</grant> <source-network> <source-address>172.0.0.0</source-address> <source-wild-card-bits>0.0.255.255</source-wild-card-bits> </source-network> </access-list-entry> </access-list-entries> </access> </accesses> </ipv4-acl-and-prefix-list> </config> </edit-config> </rpc>`  `Commit:` `<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <commit/> </rpc>`

<lock>	クライアントがデバイスの設定データストア システム全体をロックすることができます	実行中の設定をロックします。
		`Request:`
		`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <lock> <target> <running/> </target> </lock> </rpc>`
		
		`Response :`
		`<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
		
		`<ok/> </rpc-reply>`

NETCONF 動作	説明	例
<Unlock>	以前にロックされた設定をリリースします。 次のいずれかの条件が true の場合、<unlock> 操作は失敗します。 • 指定されたロックが現在アクティブではない。 • <unlock> 操作を発行するセッションが、ロックを取得したセッションと同じではない。	同一セッションの実行中の設定をロックおよびロック解除します。 Request: rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <unlock> <target> <running/> </target> </unlock> </rpc> Response - <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>
<close-session>	セッションを終了します。サーバは、セッションに関連付けられているロックとリソースをリリースし、関連付けられた接続を終了します。	NETCONF セッションを終了します。 Request : <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <close-session/> </rpc> Response: <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>
<kill-session>	現在処理中の操作を中止し、そのセッションに関連付けられているロックとリソースをリリースして、関連付けられた接続を終了します。	ID が他のセッション ID の場合は、セッションを中止します。 Request: <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <kill-session> <session-id>4</session-id> </kill-session> </rpc> Response: <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>

設定を取得する NETCONF 操作

次に、CDP 機能に対する NETCONF <get-config> 要求の動作の例を示します。

クライアントはメッセージを送信し、ルータ上で実行している CDP の現在の設定を取得します。ルータは現在の CDP 設定で応答します。

NETCONF 要求（クライアントからルータへ）	Netconf 応答（ルータからクライアントへ）
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get-config> <source><running/></source> <filter> <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg"/> </filter> </get-config> </rpc>	<?xml version="1.0"?> <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg"> <timer>10</timer> <enable>true</enable> <log-adjacency></log-adjacency> <hold-time>200</hold-time> <advertise-v1-only></advertise-v1-only> </cdp> #22 </data> </rpc-reply>

要求および応答メッセージ内の <rpc> 要素は、クライアントとルータ間で送信される NETCONF 要求を囲みます。<rpc> 要素内の `message-id` 属性は必須です。この属性は送信者によって選択された文字列で、整数をエンコードします。<rpc> 要素の受信者はこの文字列を復号化したり解釈したりしせず、<rpc-reply> メッセージ内で使用するために保存するだけです。送信者は `message-id` 値が正規化されていることを確認する必要があります。クライアントがサーバから情報を受信した場合、<rpc-reply> メッセージには同じ `message-id` が含まれています。

NETCONFセッションでのデータモデルを使用したルータクロックの設定

NETCONF は、ネットワークを設定するためにセキュア シェル（SSH）転送で使用される XML ベースのプロトコルです。クライアントアプリケーションはこのプロトコルを使用してルータの情報を要求し、ルータの設定を変更します。

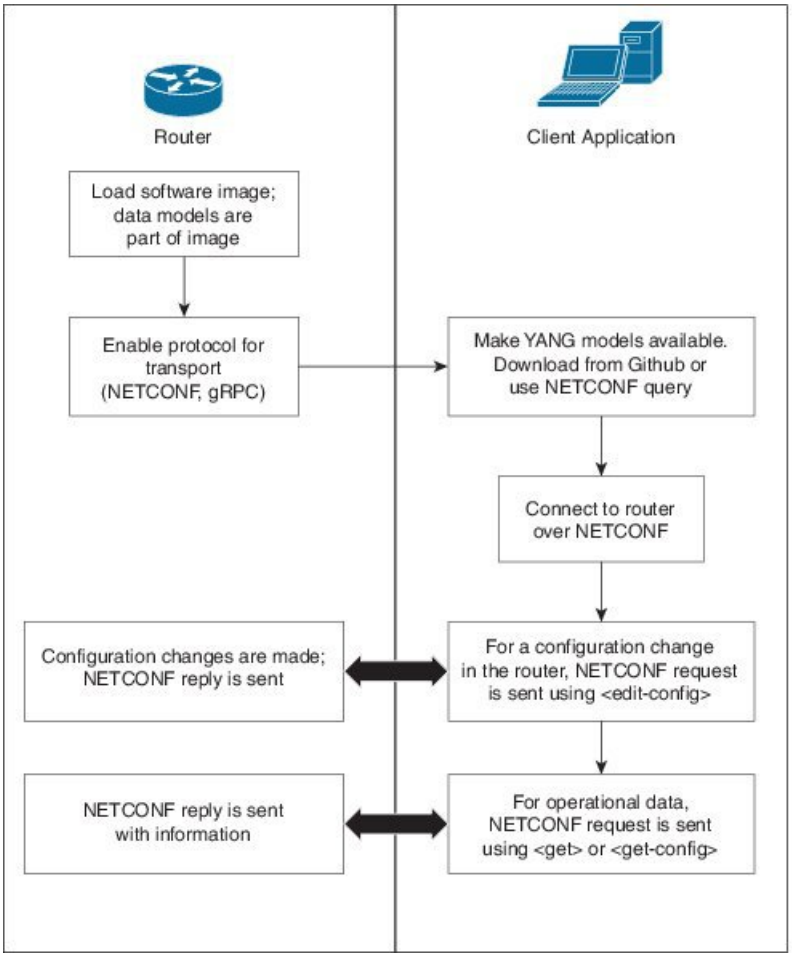
データ モデルを使用するプロセス：

- データ モデルを取得します。
- NETCONF 通信プロトコルを使用して、ルータとクライアント間の接続を確立します。
- データ モデルを使用してクライアントからルータの設定を管理します。

（注） ユーザが制御されていないアクセスを行うのを制限するために AAA 認証を設定します。AAA 認証が設定されていない場合、ユーザに割り当てられたグループに関連付けられたコマンドおよびデータ ルールはバイパスされます。IOS-XR ユーザは、ネットワーク設定プロトコル（NETCONF）、Google 定義のリモート プロシージャ コール（gRPC）または任意の YANG ベースのエージェントを介して、IOS-XR 設定への完全な読み取り/書き込みアクセス権を持つことができます。制御されていないアクセスを許可しないようにするには、いずれかの設定を行う前に AAA 認証を有効にします。

次の図は、データ モデルの使用に関連するタスクを示しています。

図 *2:* データ モデルを使用するプロセス



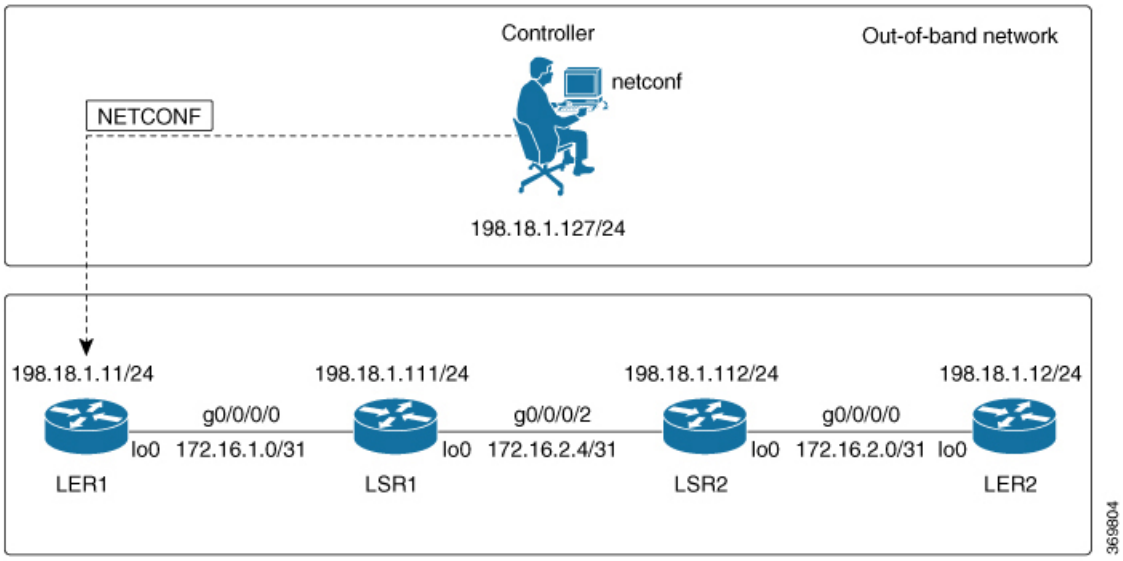
このセクションでは、ネイティブデータモデルを使用してルータクロックを設定し、NETCONF セッションを使用してクロック状態を確認します。

4 台のルータと 1 台のコントローラで構成されるネットワーク トポロジを考えてみます。ネットワークは、ラベル エッジ ルータ（ルータ）とラベル スイッチング ルータ（LSR）で構成されています。LER1 および LER2 の 2 つのルータはラベル エッジ ルータであり、2 つのルータ

LSR1 と LSR2 はラベル スイッチング ルータです。ホストは gRPC クライアントを使用するコントローラです。コントローラはアウトオブバンドネットワークを介してすべてのルータと通信します。LER1 を除くすべてのルータは、適切な IP アドレッシングとルーティングの動作によって事前に設定されています。ルータ間のインターフェイスには、`/31` アドレッシングを使用したポイントツーポイント設定があります。ループバック プレフィックスは `172.16.255.x/32` 形式を使用します。

次の図に、ネットワーク トポロジを示します。

図 *3: gRPC* セッションのネットワーク トポロジ



Cisco IOS XR ネイティブ モデルの `Cisco-IOS-XR-infra-clock-linux-cfg.yang` および `Cisco-IOX-XR-shellutil-oper` を使用すると、ルータ クロックをプログラムで設定できます。pyang などの YANG バリデータ ツールを使用して、データ モデルの構造を調べることができます。

始める前に

NETCONF モニタリング RPC を使用して、ルータ上の YANG モジュールのリストを取得します。詳細については、データ モデルへのアクセスを参照してください。

ルータ クロックの設定

手順

ステップ 1 システムのローカル タイムゾーンのネイティブ設定モデルを確認します。

例：

```
controller:netconf$ pyang --format tree Cisco-IOS-XR-infra-infra-clock-linux-cfg.yang
```

```
module: Cisco-IOS-XR-infra-infra-clock-linux-cfg
    +--rw clock
        +--rw time-zone!
        +--rw time-zone-name string
        +--rw area-name string
```

ステップ 2 システム時刻のネイティブ動作状態モデルを確認します。

例：

```
controller:netconf$ pyang --format tree Cisco-IOS-XR-shellutil-oper.yang
module: Cisco-IOS-XR-shellutil-oper
    +--ro system-time
        +--ro clock
        | +--ro year? uint16
        | +--ro month? uint8
        | +--ro day? uint8
        | +--ro hour? uint8
        | +--ro minute? uint8
        | +--ro second? uint8
        | +--ro millisecond? uint16
        | +--ro wday? uint16
        | +--ro time-zone? string
        | +--ro time-source? Time-source
        +--ro uptime
            +--ro host-name? string
            +--ro uptime? uint32
```

ステップ 3 ルータ LER1 の現在の時刻を取得します。

例：

```
controller:netconf$ more xr-system-time-oper.xml <system-time
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-oper"/>
controller:netconf$ netconf get --filter xr-system-time-oper.xml
198.18.1.11:830
<?xml version="1.0" ?>
<system-time xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-oper">
    <clock>
        <year>2019</year>
        <month>8</month>
        <day>22</day>
        <hour>17</hour>
        <minute>30</minute>
        <second>37</second>
        <millisecond>690</millisecond>
        <wday>1</wday>
        <time-zone>UTC</time-zone>
        <time-source>calendar</time-source>
    </clock>
    <uptime>
        <host-name>ler1</host-name>
        <uptime>851237</uptime>
    </uptime>
</system-time>
```

タイムゾーン UTC は、ローカル タイムゾーンが設定されていないことを示していることに注意してください。

ステップ 4 太平洋標準時（PST）を LER1 のローカル タイムゾーンとして設定します。

例：

```
controller:netconf$ more xr-system-time-oper.xml <system-time
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-oper"/>
controller:netconf$ get --filter xr-system-time-oper.xml
<username>:<password>@198.18.1.11:830
<?xml version="1.0" ?>
  <system-time xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-oper">
    <clock>
        <year>2019</year>
        <month>8</month>
        <day>22</day>
        <hour>9</hour>
        <minute>52</minute>
        <second>10</second>
        <millisecond>134</millisecond>
        <wday>1</wday>
        <time-zone>PST</time-zone>
        <time-source>calendar</time-source>
    </clock>
    <uptime>
        <host-name>ler1</host-name>
        <uptime>852530</uptime>
    </uptime>
</system-time>
```

ルータ クロックの表示

ルータ クロックが PST タイムゾーンに設定されていることを確認します。

```
controller:netconf$ more xr-system-time-oper.xml
<system-time xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-oper"/>

controller:netconf$ netconf get --filter xr-system-time-oper.xml
<username>:<password>@198.18.1.11:830
<?xml version="1.0" ?>
<system-time xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-oper">
    <clock>
        <year>2018</year>
        <month>12</month>
        <day>22</day>
        <hour>9</hour>
        <minute>52</minute>
        <second>10</second>
        <millisecond>134</millisecond>
        <wday>1</wday>
        <time-zone>PST</time-zone>
        <time-source>calendar</time-source>
    </clock>
    <uptime>
        <host-name>ler1</host-name>
        <uptime>852530</uptime>
    </uptime>
</system-time>
```

つまり、ローカル タイムゾーンが設定されていないルータ LER1 は、データ モデルを使用してプログラムで設定されます。