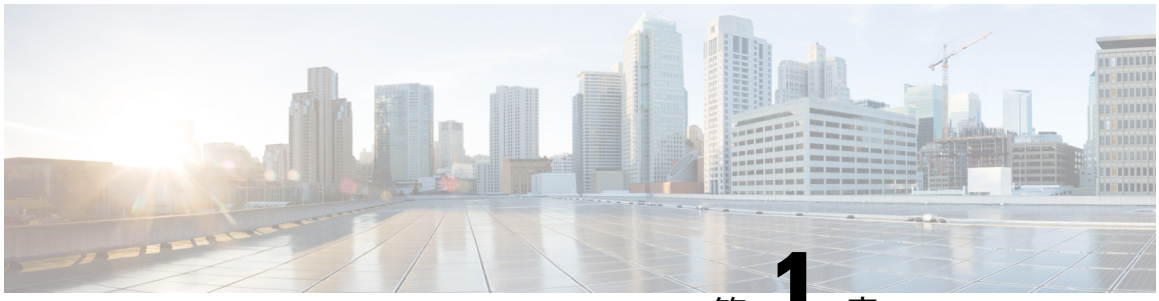
Cisco NCS 560 シリーズ ルータ（Cisco IOS XR リリース 6.6.x）プログラマビリティ コンフィギュレーション ガイド

初版：2019 年 5 月 30 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次



第 1 章

データ モデルを使用したプログラム設定

データ モデルは、ネットワーク デバイスの運用データを設定し、収集するプログラマチックな手法です。手動による設定プロセスに代わるもので、独自仕様であり、高度なテキストベースです。

データ モデル：範囲、ニーズ、および利点

スコープ

データ モデルを使用すると、ネットワーク内の異種デバイス間で設定タスクを自動化できます。

データ モデルは、ルータ（RFC 6244）で次のタイプの要件を処理します。

- **設定データ**：システムを初期のデフォルト状態から現在の状態に変えるために必要とされる、書き込み可能データのセットです。たとえば、IP ルーティング テーブルのエントリを設定したり、特定の値を使用するようにインターフェイス MTU を設定したり、イーサネット インターフェイスを特定の速度で実行するように設定したりできます。
- **運用状態データ**：実行時にシステムによって取得され、設定データと同様の方法でシステムの動作に影響を与えるデータのセットです。ただし、設定データとは対照的に、運用状態データは一時的なものです。データは、内部コンポーネントまたは特殊なプロトコルを使用する他のシステムとの相互作用によって変更されます。たとえば、OSPF のようなルーティング プロトコル、およびネットワーク インターフェイスの属性などから取得したエントリです。
- **アクション**：堅牢なネットワーク全体の設定トランザクションをサポートする NETCONF アクションのセットです。複数のデバイスに影響を与える変更が試行されると、NETCONF アクションによって障害シナリオの管理が簡略化されます。その結果、確実に成功するか、完全に失敗するトランザクションを利用できるようになります。

データ モデルでは、ルータの設定データと運用データが明確に定義された階層、および NETCONF アクションが提供されます。データ モデルは、ネットワーク全体に展開する設定の共通フレームワークを提供するようにプログラムされています。この共通フレームワークにより、ネットワークを簡単にプログラムおよび管理できます。

データ モデルの詳細については、RFC 6244 を参照してください。

必要性

通常、ネットワーク オペレーション センターには、ネットワークの複数のレイヤに多種多様なデバイスが存在しています。このようなネットワーク センターでは、設定が一括で自動化され、シームレスに実行される必要があります。

CLI は、ルータの動作の詳細を設定および抽出するために広く使用されています。ただし、CLI スクレイピングの一般的なメカニズムには柔軟性がなく、最適でもありません。設定を少し変更するには、スクリプトを複数回記述する必要があります。CLI を使用して設定を一括で変更すると、複雑になり、エラーが発生しやすくなります。このように自動化や拡張に対する制約があります。

このような制約を克服するために、Cisco IOS XR は、データ モデルを使用して任意のネットワーク デバイスにプログラムで設定を記述する方法をサポートしています。

データ モデルは、設定データの操作、運用データの取得、アクションの実行に役立ちます。データ モデルは、手動による設定のプロセスに代わるもので、業界で定義されている言語で記述されています。CLI を使用した設定のほうが簡単で判別しやすいですが、データ モデルを使用して設定を自動化すると拡張性が得られます。

データ モデルでは、ネットワーク設定プロトコル（NETCONF）または gRPC（google で定義されたリモート プロシージャ コール）プロトコルを使用してネットワーク内のデバイスの機能にアクセスできます。ルータ上の操作は YANG モデルを使用してプロトコルによって実行され、ネットワーク内の操作を自動化およびプログラムします。

ネットワーク内の設定を自動化するプロセスは、コア コンポーネント（ルータ、クライアント アプリケーション、YANG モデル、および通信プロトコル）を使用して実現されます。

利点

データ モデルを使用したルータ設定では、従来のルータの管理がもたらす欠点が解消されます。これはデータ モデルで次のことが行われるためです。

- 設定データと運用状態データに共通するモデルを提供し、NETCONF アクションを実行します。
- プロトコルを使用してルータと通信し、ネットワーク内の設定を取得、操作、および削除します。
- ネットワーク全体の複数のルータの設定と操作を自動化します。

データ モデルを使用するプロセス

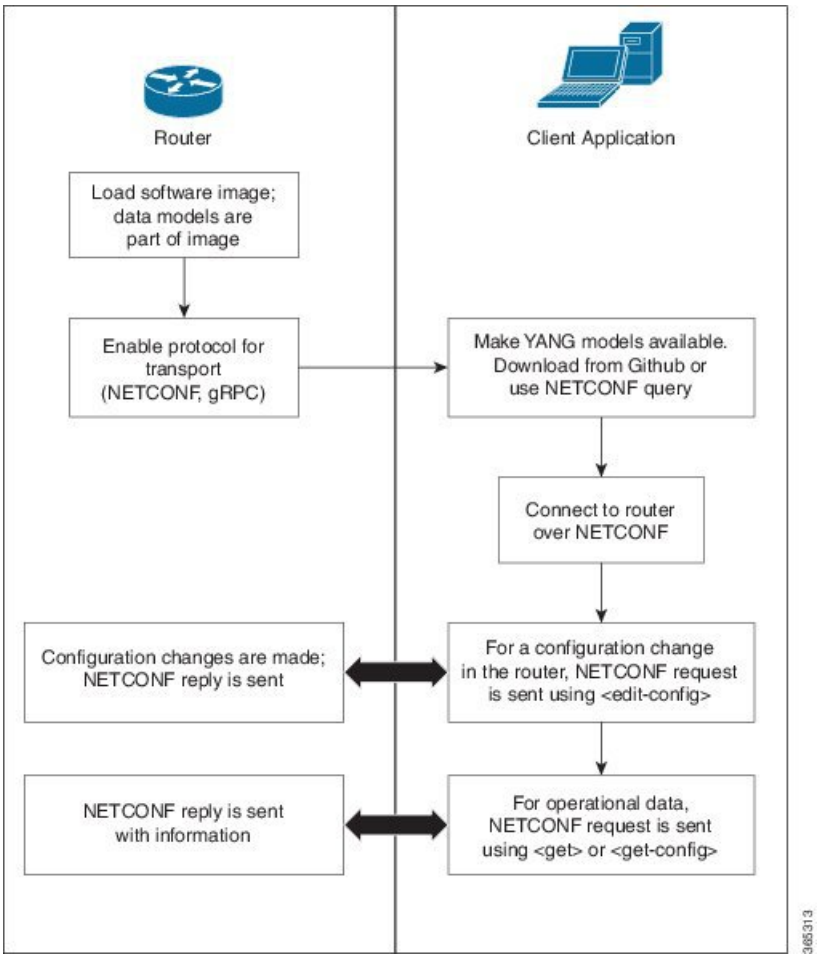
データ モデルを使用するプロセス：

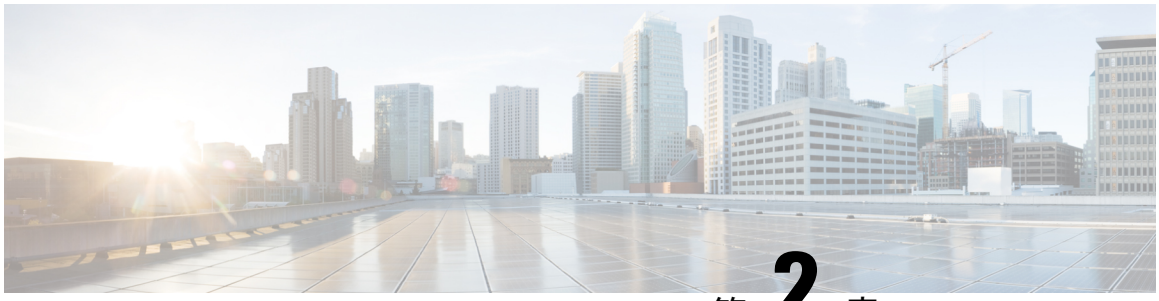
- データ モデルを取得します。
- NETCONF または gRPC などの通信プロトコルを使用して、ルータとクライアント間の接続を確立します。
- データ モデルを使用してクライアントからルータの設定を管理します。

（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。

図 1 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 *1:* データ モデルを使用するプロセス





第 2 章

データ モデルの使用

データ モデルの使用には、次の 3 つのタスクがあります。

データ モデルの取得

データ モデルは、MGBL PIE のソフトウェア パッケージから入手できます。ルータにパッケージをインストールすると、そのパッケージに含まれる特定の機能がインストールされます。Cisco IOS XR ソフトウェアはさまざまなソフトウェア パッケージに分割されているため、ルータで実行する機能を選択することができます。各パッケージには、ルーティングやセキュリティなど、特定のルータ機能のセットを実行するコンポーネントが含まれています。

前提条件：

MGBL PIE ソフトウェア イメージがルータにロードされていることを確認します。

インストール手順については、『*System Setup and Software Installation Guide for Cisco NCS 540 Series Routers*』の「*Perform System Upgrade and Install Feature Packages*」の章を参照してください。

1. データ モデルが `netconf-monitoring` 要求で使用可能であることを確認します。

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
 <get>
   <filter type="subtree">
     <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
     <schemas/>
     </netconf-state>
   </filter>
 </get>
</rpc>
```

すべての IOS XR およびシステム管理 YANG モデルが表示されます。

YANG モデルは、**get-schema** コマンドを使用してルータにログインしなくてもルータから取得できます。

スキーマ リストを取得します（データはステップ 2 で使用されます）。

```
<get>
<filter type="subtree">
<netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
<schemas/>
</netconf-state>
</filter>
</get>
</rpc>
```

ルータ上のすべてのモデルが表示されます。

```
TRACE: 2016/06/13 11:11:42 transport.go:104: Reading from connection
TRACE: 2016/06/13 11:11:42 gnc_main.go:587: Session established (Id: 1009461378)
TRACE: 2016/06/13 11:11:42 session.go:93: Request:
<rpc message-id="16a79f87-1d47-4f7a-a16a-9405e6d865b9"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><get><filter type="subtree"><netconf-state

xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring"><schemas/></netconf-state></filter></get></rpc>
TRACE: 2016/06/13 11:11:42 transport.go:104: Reading from connection
TRACE: 2016/06/13 11:11:42 session.go:117:
Response:
#143589
<rpc-reply message-id="16a79f87-1d47-4f7a-a16a-9405e6d865b9"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
<netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
<schemas>
<schema>
<identifier>Cisco-IOS-XR-crypto-sam-oper</identifier>
<version>2015-01-07</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-sam-oper</namespace>
<location>NETCONF</location>
</schema>
<schema>
<identifier>Cisco-IOS-XR-crypto-sam-oper-sub1</identifier>
<version>2015-01-07</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-crypto-sam-oper</namespace>
<location>NETCONF</location>
</schema>
<schema>
<identifier>Cisco-IOS-XR-snmp-agent-oper</identifier>
<version>2015-10-08</version>
<format>yang</format>
<namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-snmp-agent-oper</namespace>
<location>NETCONF</location>
</schema>
------------<truncated>--------------
```

データ モデルの構造の詳細については、YANG モジュール （15 ページ）を参照してください。

次の作業：

ルータとクライアント アプリケーション間の接続を確立するためのプロトコルを有効にします。

プロトコル有効

ルータはプロトコルを使用してクライアントアプリケーションと通信します。ルータおよびクライアント アプリケーションで、要件に基づいて通信プロトコルを有効にします。

- NETCONF
- gRPC

（注） XR で作成された最初の root-lr ユーザのみがシステム管理の最初の root-system ユーザとして同期されますが、連続するユーザは同期されません。XR で作成された連続するユーザはシステム管理には存在しません。そのため、NETCONF または gRPC を介して、連続するユーザによって実行される sysadmin アクセスが必要な操作は失敗します。この制約を克服するには、システム管理で同じ名前のユーザを作成し、適切なグループに割り当てることによって権限を付与します。

プロトコルの詳細については、コミュニケーション プロトコル （21 ページ）を参照してください。

SSH プロトコルを介した NETCONF の有効化

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。

NETCONF の詳細については、NETCONF プロトコル （21 ページ）を参照してください。

前提条件：

- ソフトウェア パッケージ k9sec package/rpm がルータにインストールされている。
- ソフトウェア パッケージ mgbl package/rpm がルータにインストールされている。
- 暗号キーが生成されている。

NETCONF プロトコルを有効にするには、次の手順を実行します。

1. SSH 接続を介して NETCONF プロトコルを有効にします。

```
ssh server v2
ssh server netconf
netconf agent tty
netconf-yang agent ssh
```

   デフォルトのポート番号 830 が使用されています。必要に応じて、1 ～ 65535 内の別のポートを指定できます。

2. セッション パラメータを設定します。

```
router (config)# netconf-yang agent session {  limit value |  absolute-timeout value
 |  idle-timeout value }
```

値は次のとおりです。

- **limit value**：netconf-yang 同時セッションの最大数を設定します。有効な範囲は 1 ～ 1024 です。
- **absolute-timeout value**：絶対セッション ライフタイムを分単位で設定します。指定できる範囲は 1 ～ 1440 です。
- **idle-timeout value**：アイドル セッション ライフタイムを分単位で設定します。指定できる範囲は 1 ～ 1440 です。

3. 統計情報とクライアントの構成設定を確認します。

```
router (config)# do show netconf-yang statistics
router (config)# do show netconf-yang clients
```

例：NETCONF の有効化

```
config
 netconf-yang agent ssh
  ssh server netconf port 830
!
```

例：統計情報を使用した設定の確認

NETCONF 要求の送信後、**do show netconf-yang statistics** コマンドを使用して設定を確認します。

```
show netconf-yang statistics
Summary statistics        requests|             total time|   min time per request|
max time per request|   avg time per request|
other                            0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
close-session                    4|       0h  0m  0s   3ms|       0h  0m  0s   0ms|
    0h  0m  0s   1ms|       0h  0m  0s   0ms|
kill-session                     0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
get-schema                       0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
get                              0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s
get-config                       1|       0h  0m  0s   1ms|       0h  0m  0s   1ms|
    0h  0m  0s   1ms|       0h  0m  0s   1ms|
edit-config                      3|       0h  0m  0s   2ms|       0h  0m  0s   0ms|
    0h  0m  0s   1ms|       0h  0m  0s   0ms|
commit                           0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
cancel-commit                    0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
lock                             0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
unlock                           0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
```

```
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
discard-changes                    0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
validate                           0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
    0h  0m  0s   0ms|       0h  0m  0s   0ms|
```

例：クライアントを使用した設定の確認

```
show netconf-yang clients
client session ID|   NC version|    client connect time|        last OP time|
last OP type|    <lock>|
22969|                 1.1|         0d  0h  0m  2s|             11:11:24|
close-session|        No|
15389|
```

次の作業：

NETCONF を有効にした後、YANG データ モデルを使用して関連する設定を管理します。

HTTP/2 プロトコルを介した gRPC の有効化

Google 定義されたリモート プロシージャ コール（gRPC）は、オープンソースの RPC フレームワークです。gRPC は IPv4 および v6 アドレス ファミリをサポートしています。

gRPC の詳細については、gRPC プロトコル （26 ページ）を参照してください。

前提条件：

- TLS を設定する。

（注） TLS を設定することをお勧めします。gRPC プロトコルを有効にすると、TCP で TLS が有効になっていないデフォルトの HTTP/2 トランスポートが使用されます。gRPC では、すべての gRPC 要求に対して AAA 認証および認可が義務付けられています。TLS が設定されていない場合、認証クレデンシャルはネットワーク上で暗号化されずに転送されます。TLS を有効にすると、クレデンシャルがセキュアで暗号化されることが保証されます。非 TLS モードは、セキュアな内部ネットワークでのみ使用できます。

- ソフトウェア パッケージ mgbl pie がルータにインストールされている。

gRPC プロトコルを有効にするには、次の手順を実行します。

1. HTTP/2 接続で gRPC を有効にします。

```
Router# configure
Router (config)# grpc
```

2. 指定されたポート番号へのアクセスを有効にします。

```
Router (config-grpc)# port <port-number>
```

<port-number> の範囲は 57344 ～ 57999 です。ポート番号が使用できない場合は、エラーが表示されます。

3. コンフィギュレーション モードでセッション パラメータを設定します。

```
Router (config)# grpc{ address-family | dscp | max-request-per-user | max-request-total
 | max-streams | max-streams-per-user | no-tls | service-layer | tls-cipher |
tls-mutual | tls-trustpoint | vrf }
```

値は次のとおりです。

- **address-family**：アドレス ファミリ識別子タイプを設定します
- **dscp**：送信された gRPC での QOS マーキング DSCP を設定します
- **max-request-per-user**：ユーザあたりの同時要求の最大数を設定します
- **max-request-total**：合計同時要求の最大数を設定します
- **max-streams**：同時 gRPC 要求の最大数を設定します。サブスクリプションの上限は 128 要求です。デフォルトは 32 要求です
- **max-streams-per-user**：ユーザあたりの同時 gRPC 要求の最大数を設定します。サブスクリプションの上限は 128 要求です。デフォルトは 32 要求です
- **no-tls**：トランスポート レイヤ セキュリティ（TLS）を無効化します。TLS はデフォルトで有効になっています。
- **service-layer**：gRPC サービス レイヤの設定を有効にします
- **tls-cipher**：gRPC TLS 暗号スイートを有効にします
- **tls-mutual**：相互認証を設定します
- **tls-trustpoint**：トラストポイントを設定します
- **server-vrf**：サーバ VRF を有効にします

次の作業：

gRPC を有効にした後、YANG データ モデルを使用して関連する設定を管理します。

データ モデルを使用した設定の管理

クライアント アプリケーションから、データ モデルを使用してルータの設定を管理します。

前提条件

- ソフトウェア パッケージ k9sec pie および mgbl がルータにインストールされている。
- NETCONF または gRPC プロトコルはクライアントとルータで有効になっている。

データ モデルを使用して設定を管理するには、次の手順を実行します。

1. YANG ツールを使用してクライアント アプリケーションにデータ モデルをインポートします。
2. YANG ツールを使用してデータ モデルの値を変更することによってルータを設定します。

設定可能なデータ モデルの値の詳細については、YANG モデルの構造 （18 ページ）を参照してください。

例：CDP の設定

この例では、CDP にデータ モデルを使用し、次の表に示す値を使用して CDP を設定しています。

CDP パラメータ	説明	パラメータに必要な値
CDPバージョン	隣接デバイスとの通信に使用するバージョンを指定します	v1
Hold time	受信デバイスが CDP パケットを保持する時間を指定します	200 ms
Timer	ソフトウェアが CDP アップデートを送信する頻度を指定します	80 ms
Log Adjacency Table	隣接関係テーブルに変更を記録します。CDP 隣接関係テーブルのロギングをイネーブルにすると、CDP ネイバーが追加または削除されるたびに syslog が生成されます。	enable

1. ルータから CDP `Cisco-IOS-XR-cdp-cfg.yang` の設定 YANG データ モデルをダウンロードします。データ モデルをダウンロードするには、データ モデルの取得 （5 ページ）を参照してください。
2. 任意の YANG ツールを使用してクライアント アプリケーションにデータ モデルをインポートします。
3. データ モデルのリーフ ノードを変更します。
   - enable（cdp をイネーブルにする）
   - holdtime
   - timer
   - advertise v1 のみ

- log adjacency

NETCONF を使用した CDP の設定

この例では、CDP のデータ モデルを使用し、NETCONF RPC 要求を使用して CDP を設定します。

```
<edit-config>
  <target>
   <candidate/>
  </target>
  <config xmlns:xc="urn:ietf:params:xml:n:netconf:base:1.0">
   <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg">
   <timer>80</timer>
   <enable>true</enable>
   <log-adjacency></log-adjacency>
   <hold-time>200</holdtime>
   <advertise-v1-only></advertise-v1-only>
  </cdp>
 </config>
</edit-config>
```

（注） また、CDPはインターフェイスマネージャを拡張してインターフェイス設定でも設定できます。`Cisco-IOS-XR-ifmgr-cfg` YANG モデルを使用し、インターフェイス設定で CDP を設定します。

gRPC を使用した CDP の設定

この例では、CDP のデータ モデルを使用し、gRPC MergeConfig RPC 要求を使用して CDP を設定します。

```
{
 "Cisco-IOS-XR-cdp-cfg:cdp": {
  "timer": 50,
  "enable": true,
  "log-adjacency": [
   null
  ],
  "hold-time": 180,
  "advertise-v1-only": [
   null
  ]
 }
}
```

（注） また、CDPはインターフェイスマネージャを拡張してインターフェイス設定でも設定できます。`Cisco-IOS-XR-ifmgr-cfg` YANG モデルを使用し、インターフェイス設定で CDP を設定します。

設定のコミット

現在実行中の設定で新しい値を設定するには、設定をコミットします。

また、confimed-commit 操作を介して設定をコミットすることもできます。NETCONF および gRPC は confirmed-commit RPC をサポートしています。この RPC では、設定がルータで有効になる前にユーザが明示的に確認する必要があります。この機能は、設定の変更が正しく適用されていることを確認するのに役立ち、管理接続に変更は起こりません。設定変更によって管理接続が失われると、600 秒のデフォルトの confirm-timeout 期間後に、設定が以前コミットした設定に自動的にロールバックされます。

設定をコミットするには、</commit> RPC を使用します。

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

設定に対して confirm-commit を実行する場合は、次のとおりです。

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

confirmed-commit 機能は、<cancel-commit> 操作、<commit> 操作の <confirmed>、<confirm-timeout>、<persist>、<persist-id> パラメータをサポートしています。

```
<rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <commit>
         <confirmed/>
         <persist>IQ,d4668</persist>
         <confirm-timeout>120</confirm-timeout>
       </commit>
</rpc>
```

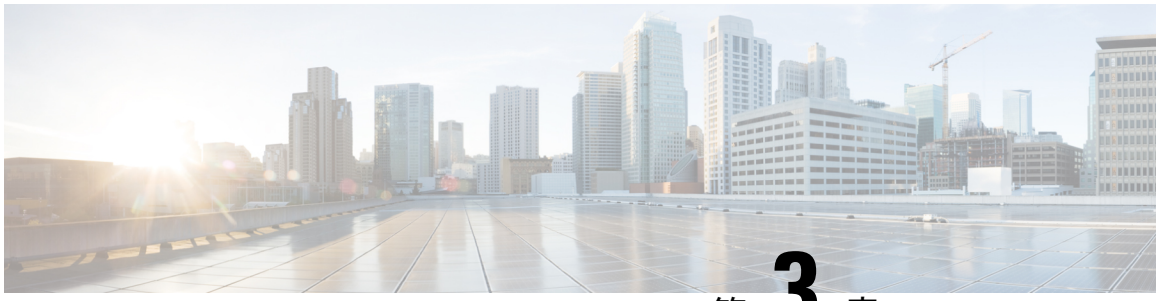
confirmed-commit 要求は、次の場合に Datastore Locked エラーが発生して失敗します。

- confirmed-commit 操作と confirming-commit 操作の間で別の操作を実行する場合
- 別のセッションでアクティブな confirmed-commit 要求があり、永続 ID が指定されていない場合
- 永続 ID は指定されているが、アクティブな confirmed-commit セッションの永続 ID と一致しない場合

設定をマージする gRPC Confirmed-Commit

gRPC confirmed-commit 要求は、既存の merge-config 操作および cli-config 操作に対して発行できます。この例では、要求は merge-cli 操作に対して作成されています。

```
manageability/ems/client/client -oper merge-config -server_addr="<address>" -json_in_file
 <directory-path>/<file>.json
-confirmed=yes -confirm_timeout=400
enter PID:14917:main.main
emsMergeConfig: Received ReqId 14917, Response '
----------------- gRPC Summary ----------------------
Operation: merge-config
Number of iterations: 1
Total bytes transferred: 126
Number of bytes per second: 374
Round trip throughputs Mbps: 0.002999
Ave elapsed time in seconds: 0.336079
Min elapsed time in seconds: 0.336079
Max elapsed time in seconds: 0.336079
--------------- End gRPC Summary ----------------------
The confirmed commit request should be followed by a confirming commit to make the
configuration permanent:
manageability/ems/client/client -oper commit -server_addr="<address>"
enter PID:14917:main.main
emsCommitConfig: Received ReqId 14917, Response '
----------------- gRPC Summary ----------------------
Operation: commit
Number of iterations: 1
Total bytes transferred: 126
Number of bytes per second: 374
Round trip throughputs Mbps: 0.002999
Ave elapsed time in seconds: 0.336079
Min elapsed time in seconds: 0.336079
Max elapsed time in seconds: 0.336079
--------------- End gRPC Summary ----------------------
```



第 3 章

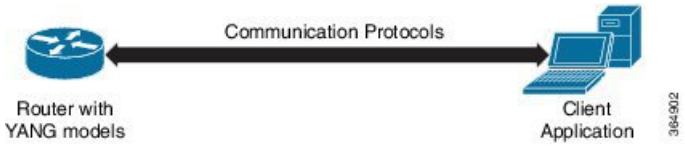
データ モデルを使用するコンポーネント

ネットワーク内の設定を自動化するプロセスでは、次のコア コンポーネントを使用します。

- **クライアント アプリケーション**：ネットワーク内のデバイスの設定を管理および監視します。
- **ルータ**：サーバとして機能し、クライアント アプリケーションからの要求に応答し、ネットワーク内のデバイスを設定します。
- **YANG モジュール**：ルータの設定データと運用データを列挙し、アクションを実行します。
- **通信プロトコル**：ネットワーク デバイスの設定をインストール、操作、削除するためのメカニズムです。

図 2 はコア コンポーネントの相互作用を示しています。

図 *2:* データ モデルを使用する際のコンポーネント



この章では、2 つの コンポーネントについて説明します。

YANG モジュール

YANG モジュールは、ルータのデータを介してデータ モデルを定義し、そのデータに対する階層的な組織と制約を定義します。各モジュールは、名前空間 URL によって一意に識別されます。YANG モデルはネットワーク デバイスの設定データと運用データ、実行アクション、リモート プロシージャ コール、および通知を記述します。

YANGモデルはルータから取得する必要があります。モデルは、ルータとクライアントの間で交換されるデータの有効な構造を定義します。モデルは NETCONF および gRPC 対応アプリケーションで使用されます。

YANG モデルは次のとおりです。

- **シスコ固有のモデル**：サポート対象モデルおよびその表記のリストについては、https://github.com/YangModels/yang/tree/master/vendor/cisco/xr/を参照してください。
- **共通モデル**：これらのモデルは、IETF や IEEE などの標準化機関の業界全体の標準 YANG モデルです。また、これらのモデルは Open Config（OC）モデルとも呼ばれます。合成モデルの場合と同様に、OC モデルには、設定データ、運用データ、アクションに対して定義された個別の YANG モデルがあります。

サポートされている OC モデルとその表記のリストについては、https://github.com/openconfig/public/tree/master/release/modelsを参照してください。

YANG の詳細については、RFC 6020 および 6087 を参照してください。

YANG モジュールのコンポーネント

YANG モジュールでは、単一のデータ モデルを定義します。ただし、モジュールは、次のいずれかのステートメントを使用して他のモジュールおよびサブモジュールで定義を参照できます。

- **import** は外部モジュールをインポートします
- **include** には 1 つ以上のサブモジュールが含まれます
- **augment** は別のモジュールを拡張し、データ モデル階層で新しいノードの配置を定義します
- **when** は新しいノードが有効な条件を定義します
- **prefix** はインポートされたモジュールの定義を参照します

YANGモデルでは、機能の設定、ルータの運用状態の取得、およびアクションの実行が行われます。

（注） gRPC YANG パスまたは JSON データは、YANG 名前空間ではなく、YANG モジュール名に基づいています。

例：AAA の設定 YANG モデル

機能の設定に使用される YANG モデルは -cfg と表記されます。

```
(snippet)
module Cisco-IOS-XR-aaa-locald-cfg {
```

```
  /*** NAMESPACE / PREFIX DEFINITION ***/

  namespace "http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-cfg";


  prefix "aaa-locald-cfg";

  /*** LINKAGE (IMPORTS / INCLUDES) ***/

  import Cisco-IOS-XR-types { prefix "xr"; }

  import Cisco-IOS-XR-aaa-lib-cfg { prefix "a1"; }

  /*** META INFORMATION ***/

  organization "Cisco Systems, Inc.";
          .......................
          ....................... (truncated)
```

例：AAA の運用 YANG モデル

運用データの取得に使用される YANG モデルは -oper と表記されます。

```
 (snippet)
module Cisco-IOS-XR-aaa-locald-oper {

  /*** NAMESPACE / PREFIX DEFINITION ***/

  namespace "http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-oper";


  prefix "aaa-locald-oper";

  /*** LINKAGE (IMPORTS / INCLUDES) ***/

  import Cisco-IOS-XR-types { prefix "xr"; }

  include Cisco-IOS-XR-aaa-locald-oper-sub1 {
    revision-date 2015-01-07;
  }

  /*** META INFORMATION ***/

  organization "Cisco Systems, Inc.";
    .......................
    ....................... (truncated)
```

（注） 1 つのモジュールにサブモジュールをいくつでも組み込むことができますが、各サブモジュールが属すことができるのは1つのモジュールのみです。すべての標準モジュールおよびサブモジュールの名前は一意である必要があります。

例：OSPFv3 の NETCONF アクション

アクションの実行に使用される YANG モデルは -act と表記されます。

```
 (snippet)
clear ospfv3 1 vrf vrf1 statistics neighbor 2.2.2.2
RPC message based on the new ospfv3 yang model-
 <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <act-ospfv3-instance-vrf xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv6-ospfv3-act">

     <instance>
       <instance-identifier>1</instance-identifier>
        <vrf>
         <vrf-name>vrf1</vrf-name>
         <stats>
          <neighbor>
           <neighbor-id>2.2.2.2</neighbor-id>
          </neighbor>
         </stats>
        </vrf>
     </instance>
   </act-ospfv3-instance-vrf>
 </rpc>
```

YANG モデルの構造

YANG データ モデルはノードのある階層型のツリーベース構造で表現できます。この表現により、モデルを簡単に理解できるようになります。

各機能には、スキーマから合成された定義済みの YANG モデルがあります。ツリー形式のモデルは次のとおりです。

- トップ レベル ノードおよびサブツリー
- 他の YANG モデル内でノードを拡張するサブツリー
- カスタム RPC

YANG は 4 つのノード タイプを定義します。各ノードには名前があります。ノード タイプに応じて、ノードは値を定義するか、一連の子ノードを含めます。データ モデリングの場合、次のノード タイプがあります。

- リーフ ノード：特定のタイプの単一の値が含まれています。
- リーフリスト ノード：一連のリーフ ノードが含まれています。
- リスト ノード：一連のリーフリスト エントリが含まれています。リーフリスト エントリのそれぞれは 1 つ以上のキー リーフによって一意に識別されます。
- コンテナ ノード：子ノードのみを含む関連ノードのグループが含まれます。子ノードは4 つのノード タイプのいずれかです。

例：CDP データ モデルの構造

Cisco Discovery Protocol（CDP）の設定には、固有の拡張モデル（インターフェイス設定）があります。この拡張は、グローバル設定レベルとインターフェイス設定レベル

の両方で CDP を設定できることを示しています。ツリー構造の CDP インターフェイス マネージャのデータ モデルは次のとおりです。

```
module: Cisco-IOS-XR-cdp-cfg
   +--rw cdp
      +--rw timer?               uint32
      +--rw advertise-v1-only?   empty
      +--rw enable?              boolean
      +--rw hold-time?           uint32
      +--rw log-adjacency?       empty
augment /a1:interface-configurations/a1:interface-configuration:
   +--rw cdp
      +--rw enable?   empty
```

CDP YANG モデルでは、拡張は次のように表現されます。

```
augment "/a1:interface-configurations/a1:interface-configuration" {
    container cdp {
      description "Interface specific CDP configuration";
      leaf enable {
        type empty;
        description "Enable or disable CDP on an interface";
      }
    }
    description
      "This augment extends the configuration data of
      'Cisco-IOS-XR-ifmgr-cfg'";
  }
```

CDP の運用上の YANG :

```
module: Cisco-IOS-XR-cdp-oper
   +--ro cdp
      +--ro nodes
         +--ro node* [node-name]
            +--ro neighbors
            |  +--ro details
            |  |  +--ro detail*
            |  |     +--ro interface-name?   xr:Interface-name
            |  |     +--ro device-id?        string
            |  |     +--ro cdp-neighbor*
            |  |        +--ro detail
            |  |        |  +--ro network-addresses
            |  |        |  |  +--ro cdp-addr-entry*
            |  |        |  |     +--ro address
            |  |        |  |        +--ro address-type?   Cdp-l3-addr-protocol
            |  |        |  |        +--ro ipv4-address?   inet:ipv4-address
            |  |        |  |        +--ro ipv6-address?   In6-addr
            |  |        |  +--ro protocol-hello-list
            |  |        |  |  +--ro cdp-prot-hello-entry*
            |  |        |  |     +--ro hello-message?   yang:hex-string
            |  |        |  +--ro version?               string
            |  |        |  +--ro vtp-domain?            string
            |  |        |  +--ro native-vlan?           uint32
            |  |        |  +--ro duplex?                Cdp-duplex
            |  |        |  +--ro system-name?           string
            |  |        +--ro receiving-interface-name?   xr:Interface-name
            |  |        +--ro device-id?                  string
            |  |        +--ro port-id?                    string
            |  |        +--ro header-version?             uint8
            |  |        +--ro hold-time?                  uint16
            |  |        +--ro capabilities?               string
            |  |        +--ro platform?                   string
```

```
.............................................. (truncated)
```

ACL YANG モデルの使いやすさの向上

この機能は、YANG モデルのユーザビリティに影響を与えるネイティブ ACL YANG モデルで特定されたいくつかの問題に対応します。次のネイティブ ACL YANG モデルでは、利便性と標準規格への準拠が改善されています。

- Cisco-IOS-XR-es-acl-cfg
- Cisco-IOS-XR-ipv4-acl-cfg
- Cisco-IOS-XR-ipv6-acl-cfg

この機能拡張の一部として、次の問題が解決されます。

- 改訂の日付と説明の不足

  問題：改訂が ACL モデルの YANG ファイルで変更されると、新しい改訂に関連する変更が正しく記述されない。

  解決策：以前のバージョンの情報を削除しなくても、以降の各バージョンで行った変更の説明が含まれるようになりました。

  Cisco-IOS-XR-ipv4-acl-cfg.yang の例：

```
 revision "2018-04-03" {
    description
      "6.5.1 revision. Correct enum value for Next-hop-type.";
  }

  revision "2018-03-23" {
    description
      "6.5.1 revision. Removing none-next-type.";
  }
```

- 文字列の無制限の使用

  問題：ACL ネイティブ モデルは *string* 型として定義されているリーフを使用するけれども、文字列の長さが定義されていないか、正しくない。

  解決策：パターンと長さのチェックが *string* 型を使用するリーフに追加されました。これにより、NETCONF は、コミット時に ACL の検証に依存するのではなく、これらのチェックを制御できます。

  次の例では、文字列の長さは 255 文字に制限され、英数字のみが許可されています。

```
typedef my-base-str-type { type string {
length "1..255";
pattern "[0-9a-fA-F]*";
} }
```

- 適切な説明文の不足

問題：ネイティブ ACL モデルのほとんどのリーフには、適切な説明が入力されていない説明フィールドがあるため、YANG モデルの使いやすさと理解に影響している。

解決策：リーフの説明フィールドが更新され、有用な情報を提供するようになりました。

- ACL 範囲リーフ間での検証の不足と一貫性のない動作

  問題：YANGモデルでは、上限値、下限値、および演算子リーフが指定されたコンテナでさまざまなリーフの組み合わせをサポートしている。その結果、CLIの観点から一部の設定が無効になる場合がある。

  解決策：ネイティブ ACL モデルは、YANG モデルでサポートされているさまざまなリーフの組み合わせを含めるように改善されました。また、すべての ACL 範囲のリーフ間の解析動作が一貫性を維持できるように配置されています。

- プロトコル演算子の入力と出力間の不整合

  問題：NETCONF 出力がユーザ入力と一致する必要がある。

  解決策：プロトコル演算子リーフを任意のものとするように変更し、*equal* に設定する必要はなくなりました。

コミュニケーション プロトコル

通信プロトコルはルータとクライアント間の接続を確立します。プロトコルにより、クライアントは YANG データ モデルを使用し、ネットワーク操作を自動化およびプログラムすることができます。

YANG は、次のいずれかのプロトコルを使用します。

- ネットワーク設定プロトコル（NETCONF）
- gRPC（google 定義リモート プロシージャ コール）

次の表に、これら 2 つのプロトコルのトランスポートおよびエンコード メカニズムを示します。

プロトコル	トランスポート	符号化/復号化
NETCONF	ssh	xml
gRPC	http/2	json

NETCONF プロトコル

NETCONF は、ネットワーク デバイスの設定をインストール、操作、または削除するためのメカニズムです。コンフィギュレーション 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。NETCONF がポート 22 経由で XML サブシステムにアクセスできるようにするには、**ssh server capability netconf-xml** コマンドを使

用します。NETCONF は、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。

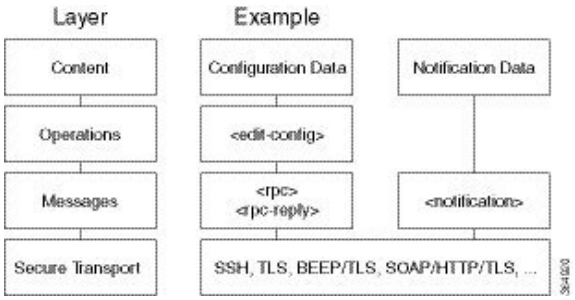
NETCONF セッション

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 は 4 つのレイヤに分割できます。

図 *3 : 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。

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
```

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>

例：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。

gRPC プロトコル

gRPC はオープンソースの RPC フレームワークです。これはプロトコル バッファ（Protobuf）に基づいたオープンソースのバイナリ シリアル化プロトコルです。gRPC は、XML などの構造化されたデータをシリアル化するための柔軟で効率的な自動メカニズムですが、小型で使いやすくなっています。ユーザは、.proto ファイルにプロトコル バッファ メッセージ タイプを定義することで構造を定義する必要があります。各プロトコル バッファ メッセージは、一連の名前と値のペアを含む情報の小型の論理レコードです。

gRPC は要求および応答をバイナリでエンコードします。gRPC は、Protobuf とともに他のコンテンツ タイプに拡張可能です。gRPC の Protobuf バイナリ データ オブジェクトは HTTP/2 を介して転送されます。

（注） gRPC を有効にする前に TLS を設定することをお勧めします。gRPC プロトコルを有効にすると、TCP で TLS が有効になっていないデフォルトの HTTP/2 トランスポートが使用されます。gRPC では、すべての gRPC 要求に対して AAA 認証および認可が義務付けられています。TLS が設定されていない場合、認証クレデンシャルはネットワーク上で暗号化されずに転送されます。非 TLS モードは、セキュアな内部ネットワークでのみ使用できます。

gRPC はクライアントとサーバ間の分散型のアプリケーションやサービスをサポートします。gRPC はサーバとクライアント間の設定データと運用データを交換するためにデバイス管理サー

ビスを構築するインフラストラクチャを提供します。そのデータの構造はYANGモデルによって定義されます。

Cisco gRPC IDL はプロトコル バッファ インターフェイス定義言語（IDL）を使用して、サービス方式を定義し、パラメータを定義し、プロトコル バッファ メッセージ タイプとしてタイプを返します。gRPC 要求はエンコードされ、JSON を使用してルータに送信されます。クライアントは、IDL で定義される RPC コールを起動してルータをプログラミングできます。

次に、gRPC 設定の proto ファイルの構文例を示します。

```
syntax = "proto3";

package IOSXRExtensibleManagabilityService;

service gRPCConfigOper {

    rpc GetConfig(ConfigGetArgs) returns(stream ConfigGetReply) {};

    rpc MergeConfig(ConfigArgs) returns(ConfigReply) {};

    rpc DeleteConfig(ConfigArgs) returns(ConfigReply) {};

    rpc ReplaceConfig(ConfigArgs) returns(ConfigReply) {};

    rpc CliConfig(CliConfigArgs) returns(CliConfigReply) {};

    rpc GetOper(GetOperArgs) returns(stream GetOperReply) {};

    rpc CommitReplace(CommitReplaceArgs) returns(CommitReplaceReply) {};
}
message ConfigGetArgs {
     int64 ReqId = 1;
     string yangpathjson = 2;
}

message ConfigGetReply {
    int64 ResReqId = 1;
    string yangjson = 2;
    string errors = 3;
}

message GetOperArgs {
     int64 ReqId = 1;
     string yangpathjson = 2;
}

message GetOperReply {
    int64 ResReqId = 1;
    string yangjson = 2;
    string errors = 3;
}

message ConfigArgs {
    int64 ReqId = 1;
    string yangjson = 2;

}

message ConfigReply {
    int64 ResReqId = 1;
    string errors = 2;
}
```

```
message CliConfigArgs {
    int64 ReqId = 1;
    string cli = 2;
}

message CliConfigReply {
    int64 ResReqId = 1;
    string errors = 2;
}

message CommitReplaceArgs {
     int64 ReqId = 1;
     string cli = 2;
     string yangjson = 3;
}

message CommitReplaceReply {
     int64 ResReqId = 1;
     string errors = 2;
}
```

gRPCExec 設定の例：

```
service gRPCExec {
    rpc ShowCmdTextOutput(ShowCmdArgs) returns(stream ShowCmdTextReply) {};
    rpc ShowCmdJSONOutput(ShowCmdArgs) returns(stream ShowCmdJSONReply) {};
    rpc ActionJSON(ActionJSONArgs) returns(stream ActionJSONReply) {};
}

message ShowCmdArgs {
     int64 ReqId = 1;
     string cli = 2;
}

message ShowCmdTextReply {
    int64 ResReqId =1;
    string output = 2;
    string errors = 3;
}

message ActionJSONArgs {
 int64 ReqId = 1;
 string yangpathjson = 2;
}

message ActionJSONReply {
 int64 ResReqId = 1;
 string yangjson = 2;
 string errors = 3;
}
```

OpenConfiggRPC 設定の例：

```
service OpenConfiggRPC {
     rpc SubscribeTelemetry(SubscribeRequest) returns (stream SubscribeResponse) {};
     rpc UnSubscribeTelemetry(CancelSubscribeReq) returns (SubscribeResponse) {};
     rpc GetModels(GetModelsInput) returns (GetModelsOutput) {};
}

message GetModelsInput {
     uint64  requestId   = 1;
     string  name        = 2;
```

```
    string  namespace   = 3;
    string  version     = 4;
    enum MODLE_REQUEST_TYPE {
         SUMMARY = 0;
         DETAIL  = 1;
     }
     MODLE_REQUEST_TYPE requestType = 5;
}

message GetModelsOutput {
     uint64  requestId   = 1;
     message ModelInfo {
         string  name        = 1;
         string  namespace   = 2;
        string  version     = 3;
         GET_MODEL_TYPE  modelType = 4;
         string modelData = 5;
     }
     repeated ModelInfo models = 2;
     OC_RPC_RESPONSE_TYPE responseCode = 3;
     string msg = 4;
}
```

gRPC の操作

gRPC の操作は次のとおりです。

gRPC の操作	説明
GetConfig	設定を取得します
GetModels	ルータで、サポートされている Yang モデルを取得します
MergeConfig	既存の設定に追加します
DeleteConfig	設定を削除します
ReplaceConfig	既存の設定の一部を変更します
CommitReplace	既存の設定を指定された新しい設定ファイルに置き換えます
GetOper	JSON を使用して運用データを取得します
CliConfig	CLI 設定を起動します
ShowCmdTextOutput	show コマンドの出力を表示します
ShowCmdJSONOutput	show コマンドの JSON 出力を表示します

例：特定のインターフェイスの設定の取得

次に、gRPC GetConfig 操作を使用して特定のインターフェイスの設定を取得する例を示します。

```
{
    "Cisco-IOS-XR-ifmgr-cfg:interface-configurations": {
        "interface-configuration": [
            {
                "active": "act",
                "interface-name": "HundredGigE0/0/1/0"
            }
        ]
    }
}
```

例：CDP コンテナの設定の削除

次に、gRPC DeleteConfig 操作で CDP コンテナとコンテナ内のリーフを削除する例を示します。DeleteConfig 引数では YANG ノードを使用してリソースを特定します。YANG ノードの値は無視され、null に設定されます。

この例では、CDP コンテナが削除されます。

```
{
"Cisco-IOS-XR-cdp-cfg:cdp": [null]
}
```

この例では、CDP コンテナ内の `hold-time` のリーフ値が削除されます。

```
{
"Cisco-IOS-XR-cdp-cfg:cdp":
{
"hold-time": [null]
}
}
```

例：CDP タイマーのマージ設定

次に、gRPC MergeConfig 操作を使用した CDP タイマーのマージ設定の例を示します。

```
{
    "Cisco-IOS-XR-cdp-cfg:cdp": {
        "timer": 50
    }
}
```

例：インターフェイスの運用データの取得

この例では、gRPC GetOper 操作を使用してインターフェイスの運用データを取得します。

```
{
    "Cisco-IOS-XR-ifmgr-oper:interface-properties": [null]
}
```

YANG アクション

IOS XR およびシステム管理アクションは、ルータ上で操作をトリガーしたり、コマンドを実行したりする RPC ステートメントです。これらのアクションは、RPC ステートメントを使用して YANG モデルとして定義されます。アクションは、対応する NETCONF RPC または gRPC 要求をルータが受信したときに実行されます。ルータがアクションを実行すると、NETCONF RPC または gRPC 応答で返信します。

たとえば、**ping** コマンドはサポートされているアクションです。つまり、YANG モデルは RPC ステートメントを使用して **ping** コマンドに対して定義されます。このコマンドは、対応する NETCONF RPC または gRPC 要求を開始することによってルータ上で実行できます。

（注） NETCONF は XML 形式をサポートし、gRPC は JSON 形式をサポートしています。

サポートされているアクションのリストについては、次の表を参照してください。

アクション	YANG モデル
logmsg	Cisco-IOS-XR-syslog-act
snmp	Cisco-IOS-XR-snmp-test-trap-act
rollback	Cisco-IOS-XR-cfgmgr-rollback-act
ping	Cisco-IOS-XR-ping-act Cisco-IOS-XR-ipv4-ping-act Cisco-IOS-XR-ipv6-ping-act
traceroute	Cisco-IOS-XR-traceroute-act Cisco-IOS-XR-ipv4-traceroute-act Cisco-IOS-XR-ipv6-traceroute-act
crypto	Cisco-IOS-XR-crypto-act
clear ospf	Cisco-IOS-XR-ipv4-ospf-act Cisco-IOS-XR-ipv6-ospfv3-act
clear isis	Cisco-IOS-XR-isis-act
clear bgp	Cisco-IOS-XR-ipv4-bgp-act
copy	Cisco-IOS-XR-shellutil-copy-act.yang
delete	Cisco-IOS-XR-shellutil-delete-act.yang

システムプロセス管理：プロセス（再起動）	Cisco-IOS-XR-sysmgr-act Cisco-IOS-XR-sysadmin-pm
システム プロセス管理：リロード （システム管理仮想マシン（VM）のリロード、ライン カード（LC）のリロード）	Cisco-IOS-XR-sysadmin-sm
システム プロセス管理：リロード （システム管理からリロードした IOS XR VM ノード）	Cisco-IOS-XR-sysadmin-sdr-mgr
システムプロセス管理：インストール	Cisco-IOS-XR-spirit-install-act
dumpcore	Cisco-IOS-XR-spirit-corehelper-cfg

例：PING NETCONF アクション

この使用例は、ルータで ping コマンドを実行するための IOS XR NETCONF アクション要求を示しています。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act">
  <destination>
   <destination>1.2.3.4</destination>
  </destination>
 </ping>
</rpc>
```

このセクションは、ルータからの NETCONF アクション応答を示しています。

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act">
  <ipv4>
   <destination>1.2.3.4</destination>
   <repeat-count>5</repeat-count>
   <data-size>100</data-size>
   <timeout>2</timeout>
   <pattern>0xabcd</pattern>
   <rotate-pattern>0</rotate-pattern>
   <reply-list>
    <result>!</result>
    <result>!</result>
    <result>!</result>
    <result>!</result>
    <result>!</result>
   </reply-list>
   <hits>5</hits>
   <total>5</total>
   <success-rate>100</success-rate>
   <rtt-min>1</rtt-min>
   <rtt-avg>1</rtt-avg>
   <rtt-max>1</rtt-max>
  </ipv4>
 </ping-response>
</rpc-reply>
```

例：XR プロセス再起動アクション

次に、NETCONF エージェントに送信されるプロセス再起動アクションの例を示します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <sysmgr-process-restart xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-sysmgr-act">
     <process-name>processmgr</process-name>
     <location>0/RP0/CPU0</location>
  </sysmgr-process-restart>
</rpc>
```

次に、NETCONF エージェントから受信したアクション応答の例を示します。

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

例：シャットダウン ダンパー プロセス

この使用例は、ルータでダンパープロセスをシャットダウンするためのシステム管理NETCONF アクション要求を示しています。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
   <data>
     <processes xmlns="http://www.cisco.com/ns/yang/Cisco-IOS-XR-sysadmin-pm">
       <all-locations>
         <location>0/RP0</location>
         <name>
           <proc-name>dumper</proc-name>
           <instance-id>0</instance-id>
           <proc-action>
             <do-what>shutdown</do-what>
             <user-name>root</user-name>
             <user-ip>1.2.3.4</user-ip>
           </proc-action>
         </name>
       </all-locations>
     </processes>
   </data>
</action>
</rpc>
```

このセクションは、ルータからの NETCONF アクション応答を示しています。

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <processes xmlns="http://www.cisco.com/ns/yang/Cisco-IOS-XR-sysadmin-pm">
   <all-locations>
    <location>0/RP0</location>
    <name>
     <proc-name>dumper</proc-name>
     <instance-id>0</instance-id>
     <proc-action>
      <proc-action-status>User root (1.2.3.4) requested shutdown for process dumper(0)
at 0/RP0
 'Sending signal 15 to stop process dumper(IID 0) pid=2439'</proc-action-status>
     </proc-action>
```

```
    </name>
   </all-locations>
  </processes>
 </data>
</rpc-reply>
```

例：コピー アクション

次に、copy アクションの RPC 要求と応答の例を示します。

RPC 要求：

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <copy xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-copy-act">
    <sourcename>//root:<location>/100MB.txt</sourcename>
    <destinationname>/</destinationname>
    <sourcefilesystem>ftp:</sourcefilesystem>
    <destinationfilesystem>harddisk:</destinationfilesystem>
    <destinationlocation>0/RSP1/CPU0</destinationlocation>
  </copy>
</rpc>
```

RPC 応答：

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-copy-act">Successfully
 completed copy operation</response>
</rpc-reply>

8.261830565s elapsed
```

例：削除アクション

次に、delete アクションの RPC 要求と応答の例を示します。

RPC 要求：

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<delete xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-delete-act">
    <name>harddisk:/netconf.txt</name>
  </delete>
</rpc>
```

RPC 応答：

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-shellutil-delete-act">Successfully
 completed delete operation</response>
</rpc-reply>

395.099948ms elapsed
```

例：インストール アクション

次に、NETCONF エージェントに送信されるインストール アクション要求の例を示します。

```
  <install-add xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spirit-install-act">
      <packagepath>/nobackup/hanaik/yang_project/img-xrv9k</packagepath>
      <packagename>xrv9k-mpls-2.1.0.0-r64102I.x86_64.rpm</packagename>
  </install-add>
```

次に、NETCONF エージェントから受信したインストール アクション応答の例を示します。

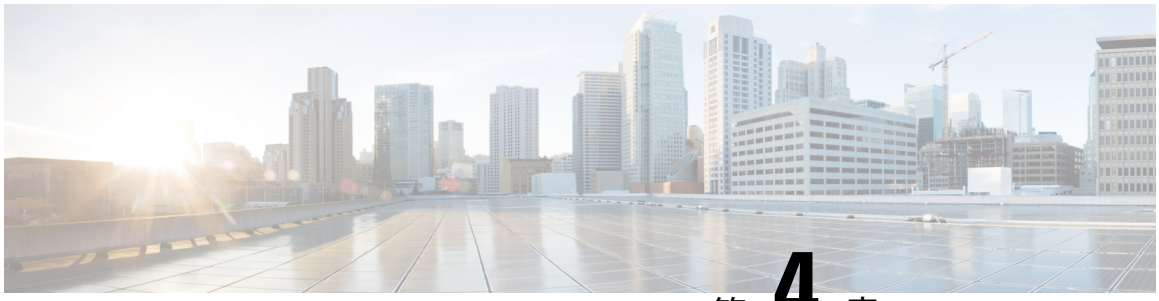
```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <op-id xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spirit-install-act">6</op-id>
</rpc-reply>
```

次に、*packagename* タグ内で囲まれた複数のパッケージを使用して *install add rpc* 要求を使用する例を示します。

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<install-add xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spirit-install-act">
   <packagepath>http://10.105.227.154/install_repo/fretta/651/651_02</packagepath>
        <packagename>ncs540-k9sec-1.0.0.0-r632.x86_64.rpm</packagename>
        <packagename>ncs540-li-1.0.0.0-r632.x86_64.rpm</packagename>
        <packagename>ncs540-mcast-1.0.0.0-r632.x86_64.rpm</packagename>
        <packagename>ncs540-mini-x.iso-6.3.2</packagename>
        <packagename>ncs540-mpls-1.0.0.0-r632.x86_64.rpm</packagename>
</install-add>
</rpc>
```

インストール アクションの制約事項

- **Install upgrade** コマンドは廃止されました。そのため、**install upgrade** コマンドの代わりに **install update** コマンドを使用します。
- 一度に送信できる要求は 1 つだけです。
- ISSU はサポートされていません。
- NETCONF アクションを使用した Yang のインストールでは、最大 32 の入力パラメータを受け入れることができます。入力パラメータは、追加、有効化、無効化、または削除するパッケージ名およびこの操作に関連する特定のログを取得するための操作 ID など、**install action** コマンドで使用される入力値です。



第 4 章

データ モデルの使用例

このセクションでは、データ モデルの特定の使用例について説明します。

BGP ネイバーの削除

この使用例では、YANG モデルを使用して BGP ネイバーを削除します。

1. 標準の YANG ツールを使用して、YANG 形式で NETCONF <get-config> 操作を使用して設定を取得します。

```
<get-config>
    <source>
      <running/>
    </source>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <bgp xmlns="http://openconfig.net/yang/bgp">
 </get-config>

router bgp 1000
 bgp confederation peers
  65002
 !
 bgp confederation identifier 102
 bgp router-id 1.1.1.1
 bgp graceful-restart restart-time 30
 bgp graceful-restart stalepath-time 30
 bgp graceful-restart
 address-family ipv4 unicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
 address-family ipv4 multicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
```

```
 !
 address-family ipv6 unicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
 address-family ipv6 multicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
!router bgp 1000
 bgp confederation peers
  65002
 !
 bgp confederation identifier 102
 bgp router-id 1.1.1.1
 bgp graceful-restart restart-time 30
 bgp graceful-restart stalepath-time 30
 bgp graceful-restart
 address-family ipv4 unicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
 address-family ipv4 multicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
 address-family ipv6 unicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
 address-family ipv6 multicast
  distance bgp 200 20 200
  maximum-paths ebgp 30
  maximum-paths ibgp 30
 !
!
```

2. 設定 <edit-config> 操作を変更します。

```
<edit-config>
    <target>
      <candidate/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <bgp xmlns="http://openconfig.net/yang/bgp">
        <global>
          <config>
            <as xc:operation="delete">1000</as>
          </config>
        </global>
      </bgp>
    </config>
  </edit-config>
```

3. NETCONF SSH からルータに <edit-config> 要求を送信します。
4. 設定の変更が成功し、BGP ネイバーが削除されていることを確認します。

（注） BGP 設定は、gRPC GetConfig 操作を使用して取得できます。

```
{
  "bgp:bgp": [
 null
  ]
}
```

gRPC DeleteConfig を使用して BGP 設定を削除します。

```
{
    "bgp:bgp": {
        "global": {
            "config": {
                "as": [
                    null
                ]
            }
        }
    }
}
```

AAA アクセス詳細の要求

この使用例では、Calvados モデルを使用して AAA アクセスの詳細を表示します。

前提条件

- ユーザが Calvados 環境に追加されていることを確認します。ユーザが XR 環境に追加され、`root-lr` 権限を持っていても、Calvados モデルへのアクセスが拒否されるためです。
- ルータとクライアント アプリケーション間で NETCONF または gRPC 接続を確立します。

（注） gRPC YANG パスまたは JSON データは、YANG 名前空間ではなく、YANG モジュール名に基づいています。

1. 標準の YANG ツールを使用し、NETCONF <get> 操作を使用してクライアントからルータに要求を送信します。

```
[ Request ]
<get>
  <filter type="subtree">
    <aaa xmlns="http://tail-f.com/ns/aaa/1.1">
      <privileged-access xmlns="http://www.cisco.com/calvados/aaa_show"/>
    </aaa>
  </filter>
</get>
```

2. ルータからクライアントに送信された応答を確認します。

```
[ Response ]
<?xml version="1.0" encoding="UTF-8"?><data
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <aaa xmlns="http://tail-f.com/ns/aaa/1.1">
   <privileged-access xmlns="http://www.cisco.com/calvados/aaa_show">
    <shell-access>None</shell-access>
    <first-user>root</first-user>
    <first-user-change>No</first-user-change>
    <current-disaster-recovery-user>root</current-disaster-recovery-user>
   </privileged-access>
  </aaa>
 </data>
```

（注） gRPC GetOper 要求を使用してこのタスクを実行するには、次の手順を実行します。

```
{
    "tailf-aaa:aaa": {
        "aaa_show:privileged-access": [
            null
        ]
    }
}
```

gRPC GetOper 応答：

```
{
 "tailf-aaa:aaa": {
  "aaa_show:privileged-access": {
   "shell-access": "None",
   "first-user": "root",
   "first-user-change": "No",
   "current-disaster-recovery-user": "root"
  }
 }
}
```

OC モデルを使用したスタティック ルートの設定

この使用例では、OC ネットワーク インスタンスのローカル ルーティング（openconfig-ni-local-routing）データ モデルを使用し、ネクスト ホップでスタティック ルートを設定します。

1. ネクスト ホップ インターフェイスを使用してスタティック ルートを作成します。

```
<edit-config>
<target>
<candidate/>
</target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<network-instances xmlns="http://openconfig.net/yang/network-instance">
       <network-instance>
       <name>default</name>
        <protocols>
         <protocol>
```

```
           <identifier
xmlns:idx="http://openconfig.net/yang/policy-types">idx:STATIC</identifier>
           <name>DEFAULT</name>
          <config>
           <identifier
xmlns:idx="http://openconfig.net/yang/policy-types">idx:STATIC</identifier>
           <name>DEFAULT</name>
          </config>
<static-routes>
<static>
<prefix>5.5.5.0/24</prefix>
<config>
<prefix>5.5.5.0/24</prefix>
</config>
<next-hops>
<next-hop>
<index>link-1</index>
<config>
<index>link-1</index>
<next-hop>6.6.6.1</next-hop>
</config>
</next-hop>
</next-hops>
</static>
</static-routes>
    </protocol>
        </protocols>
       </network-instance>
      </network-instances>
</config>
</edit-config>
```

2. スタティック ルートの設定を確認します。

```
<get-config>
<source>
<candidate/>
</source>
<filter>
<network-instances xmlns="http://openconfig.net/yang/network-instance">
       <network-instance>
       <name>default</name>
        <protocols>
        <protocol>
          <identifier
xmlns:idx="http://openconfig.net/yang/policy-types">idx:STATIC</identifier>
          <name>DEFAULT</name>
<static-routes>
<static>
<prefix>5.5.5.0/24</prefix>
</static>
</static-routes>
        </protocol>
        </protocols>
       </network-instance>
      </network-instances>
</filter>
</get-config>
```

フレキシブル CLI 設定グループでの NETCONF の使用

柔軟なCLI設定グループを含むNETCONFプロトコルを使用する場合は、継承された設定を使用する必要があります。柔軟なCLI設定グループを含むCLI設定からNETCONFおよびYANGベースの設定に移行するには、次の手順を実行します。これらの手順を使用すると、さまざまなNETCONF操作で使用できる、デバイス上のすべての設定を取得できます。

1. 送信元 <running-inheritance/> を指定して NETCONF **get-config** 要求を送信します。

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running-inheritance
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-group-cfg"/>
        </source>
     </get-config>
</rpc>
##
```

この操作は、デバイス上に存在するすべての設定（継承または展開）を次の形式で返します。

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

2. 別のデバイスに設定を適用するには、NETCONF **edit-config** 要求を次の形式で送信します。

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