プログラム可能な YANG データ モデルを使用したネットワーク自動化の促進

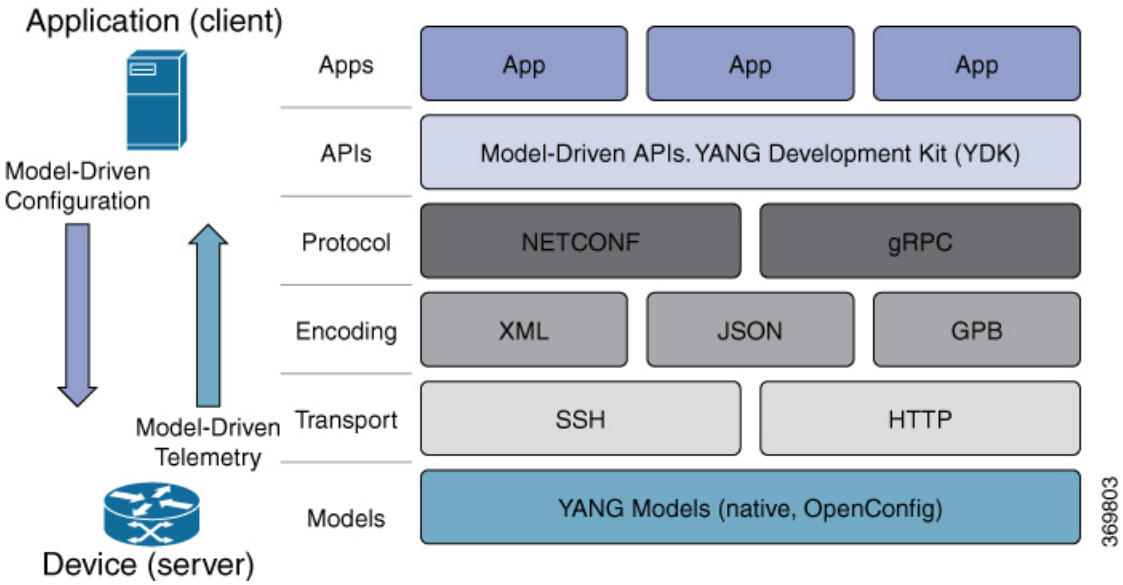
通常、ネットワーク オペレーション センターには、ネットワークの複数のレイヤに多種多様なデバイスが存在しています。このようなネットワーク センターでは、設定が一括で自動化され、シームレスに実行される必要があります。CLI は、ルータの動作の詳細を設定および抽出するために広く使用されています。ただし、CLI スクレイピングの一般的なメカニズムには柔軟性がなく、最適でもありません。設定を少し変更するには、スクリプトを複数回記述する必要があります。CLI を使用して設定を一括で変更すると、複雑になり、エラーが発生しやすくなります。このように自動化や拡張に対する制約があります。これらの制限を克服するには、ネットワークの管理用に自動化されたメカニズムが必要です。

Cisco IOS XR 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をサポートしています。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 モデルは、業界で定義された言語で記述されており、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し、運用データを取得するために使用されます。CLI を使用した設定のほうが簡単で判別しやすいですが、モデル駆動型プログラマビリティを使用して設定を自動化すると拡張性が得られます。

モデル駆動型プログラマビリティは、デバイス プログラマビリティにシンプルで柔軟性のあるフレームワークを豊富に提供します。このプログラマビリティ フレームワークは、トランスポート、プロトコル、およびエンコーディングの観点から、IOS XR デバイスとのやり取りに複数のインターフェイスを提供します。このインターフェイスは、柔軟性向上のためにモデルから分離されています。

次の図は、モデル駆動型プログラマビリティの層を示しています。

図 *1:* モデル駆動型プログラマビリティの層



データ モデルでは、ネットワーク設定プロトコル（NETCONF プロトコル）または google で定義されたリモートプロシージャ コール（gRPC 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。

データ モデルの利点

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。

ここでは、データ モデルを使用してネットワーク運用をプログラム的に管理する際の利点について説明します。

- YANG データ モデル （3 ページ）
- データ モデルへのアクセス （6 ページ）
- コミュニケーション プロトコル （7 ページ）

YANG データ モデル

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、「ネイティブ モデル」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、「OC モデル」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。

YANG データ モデルの構造

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 モデルの詳細については、RFC 6244 を参照してください。

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。

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：

データ モデルの構造は、pyang などの YANG バリデータ ツールを使用して検証され、データ モデルはツリー構造でフォーマットされます。次に、CDP運用モデルをツリー形式で表示する例を示します。

```
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

..... (snipped) .....
```

データ モデルへのアクセス

バージョン管理用のホスティング サービスを提供するソフトウェア開発プラットフォームである GitHub から Cisco IOS XR native および OpenConfig データ モデルにアクセスできます。

また、ルータからサポートされているデータ モデルにアクセスすることもできます。ルータには、データ モデルを定義する YANG ファイルが付属しています。`ietf-netconf-monitoring` 要求を使用してルータで利用可能なデータ モデルを表示するには、NETCONF プロトコルを使用します。

```
<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>
```

サポートされているすべての YANG モデルが RPC 要求に対する応答として表示されます。

```
<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>1.0.0</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>1.0.0</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>1.0.0</version>
    <format>yang</format>
    <namespace>http://cisco.com/ns/yang/Cisco-IOS-XR-snmp-agent-oper</namespace>
    <location>NETCONF</location>
</schema>

------------<snipped>--------------
<schema>
    <identifier>openconfig-aft-types</identifier>
    <version>1.0.0</version>
    <format>yang</format>
    <namespace>http://openconfig.net/yang/fib-types</namespace>
    <location>NETCONF</location>
</schema>
<schema>
    <identifier>openconfig-mpls-ldp</identifier>
    <version>1.0.0</version>
    <format>yang</format>
    <namespace>http://openconfig.net/yang/ldp</namespace>
    <location>NETCONF</location>
</schema>
</schemas>
</netconf-state>
------------<truncated>--------------
```

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。シンプルな NETCONF 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できます。NETCONF RPC の発行を開始してデータ モデルを使用してネットワーク機能を設定するには、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を参照してください。

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。`.proto` ファイルでプロトコル バッファ メッセージ タイプを定義することで、構造を定義することができます。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。NETCONF RPC の発行を開始してデータ モデルを使用してネットワーク機能を設定するには、gRPC 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を参照してください。