自分のコンピュータを設定する方法

自分のコンピュータでこのラボの作業を行うには、Putty や Terminal などの SSH クライアントが必要です。また、ネットワーク プログラマビリティのラボにアクセスできる必要もあります。

このラボの作業を行うには、事前設定済みの dCloud ラボを使用するか、自分のコンピュータを設定する必要があります。詳細については、「Pre-Event Preparation（イベント前の準備）」および「Lab Setup（ラボの設定）」の各モジュールを確認してください。

YANG およびデータ モデリングの概要

このラーニング ラボでは、YANG データ モデリング言語の基本と、この言語と NETCONF API との関係について学習します。また、Python コードを使用して dCloud ラボ環境内の CSR1000V を操作します（デバイスの NETCONF API を使用して、デバイスによってサポートされる YANG モデルを確認します）。このラーニング ラボを完了することで、YANG と NETCONF の関係を理解できるようになります。

目標

所要時間：30 分

- データ モデリングの基本を理解する
- YANG のメリットを理解する
- YANG データ モデリング言語の基本を理解する
- YANG を使用して設定および operational がモデル化されているデバイスに接続する

前提条件

- このラボでは、dCloud ラボ環境を使用します。自分のローカル ラップトップまたは Ubuntu ホストを使用してこのラボの演習を行う場合は、「Lab Setup（ラボ設定）」および「Pre-Event Preparation（イベント前の準備）」の各モジュールを見直してしてください。
- このラボの前に学習した NETCONF ラーニング ラボの内容を必ず確認してください。このラボでは、NETCONF API の操作方法に関する有益な情報が掲載されています。

背景情報

- Python に詳しくない場合には、「REST APIs and Python（REST API および Python）」モジュールを必ず確認してください。このモジュールでは、Python の基本について説明しています。

仮想環境

- このラボでは、既存の Python 環境で生じる問題を回避するために、`venv` を使用して `pyang` を実行します。
- 事前準備について確認するには、「`Lab Setup`(**ラボの設定**)」および「`Pre-Event Preparation`(**イベント前の準備**)」モジュールを参照してください。

Pyang

- 自分のコンピュータで実行する場合は、YANG データ モデルを解析するためのツールである `pyang` Python パッケージをインストールする必要があります。

ステップ 1:データ モデリングの基本を理解する

データ モデリングとは、システムやソフトウェアが使用するデータの規則や定義を説明することです。データ モデリングの詳細と、ネットワーキングのプロフェッショナルにとっての意義については、RFC3444 にデータ モデルの要件についての説明があります。データ モデリングになじみのない方がいるかもしれませんが、エンジニアであれば誰でも、この概念をよく理解しているはずです。ネットワーク エンジニアの方は、以下のような例を見たことがあるでしょう。あなたはサービス プロバイダー向けに仕事をしており、新しい顧客のインターフェイスを作成しなければならないとします。IOS で、あなたは以下のコマンドの使用を検討します。

```
interface GigabitEthernet0/1
 ip address 10.0.0.1 255.255.255.252
```

しかし、この設定を適用する際に、誤ったネットマスクを入力してしまったとします。以下の例の中に、使用できない文字が 1 字あります。

```
ISP1(config-if)#ip address 10.0.0.1 255.255.255.25w
                                                   ^
% Invalid input detected at '^' marker.
```

上記の例では IOS(ネットマスクで特定のシンタックス/構造が使用されることを理解している)がエラーをスローします。ここで、データ モデリングの演習で習得した知識が役に立ちそうです。データ モデリングを使用することで、エンジニアは入力データの正確性を確保できます。この特定の例では、デバイスが予期しない動作をするのを避けるため、無効な IP アドレスおよびネットマスクを使用した設定は適用しないようにします。データ モデリングを使用することで、システムによって受け入れられるデータのルールを定義し、適用することができます。

それでは、開発者向けの例を見てみましょう。以下に示すのは、POST operation の JSON スキーマのコード スニペットの例で、APIC-EM コントローラのポリシーを作成します。

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "description": "Policy Object",
  "defaultValue": "",
  "items": {
    "properties": {
      "id": {
        "type": "string",
        "description": "id"
      },
      "resource": {
        "properties": {
          "userIdentifiers": {
            "type": "array",
            "items": {
              "type": "string"
            },
            "uniqueItems": true
          },
          "deviceTypes": {
            "type": "array",
            "items": {
              "type": "string"
            },
            "uniqueItems": true
          }
        }
      }
    }
  }
}
```

この JSON スキーマでは、API コールのさまざまなルールやシンタックスが提供されています。たとえば、`deviceTypes` では、各種文字列を配列します。この JSON スキーマの作成者は、APIC-EM API に必要とされるデータの正確性を理解するのに役立つデータ モデルの例を提供してくれています。

この例が、データ モデリングの概要の理解に役立つよう願っています。演習の次のパートでは、YANG データ モデルのメリットの概要と、NETCONF との関係について説明します。

ステップ 2：YANG および YANG と NETCONF の関係について理解する

YANG データ モデリング言語はネットワーク ベンダーの間で一般的になり、データ モデリングのさまざまな使用例で使用されるようになりました。YANG データ モデリング言語自体には、C 言語に似たシンタックスを利用して、ネットワーク デバイスの設定および動作データをモデル化するという利点があります。YANG の基礎を学んだエンジニアであれば、ベンダーやオペレーティング システムを問わずに、YANG でモデル化されたデータを使用する RESTCONF および NETCONF API に対してプログラミングを行う方法を理解できるでしょう。

YANG は、ネットワーク エンジニアやネットワーク インフラを運用する開発者にとって非常に便利です。あらゆるベンダーのネットワーク機器によってサポートされるデータ モデルを表示するための、共通シンタックスを使用できるからです。また、XML、XPath、および何らかの基本的なプログラミングに精通している人であれば、YANG の強力なシンタックスを素早く理解できます。このことを念頭に置いて、YANG についてさらに詳しく見ていきましょう。

YANG データ モデルとは実際には何なのか

YANG データ モデルは、一般にはネットワーク サービスまたはネットワーク デバイスに対して、基本的ないくつかの動作および設定データのシンタックスおよび有効な値を定義します。

例として `ietf-inet-types.yang` データ モデルを確認してみましょう。この YANG モジュールには、すべてのネットワーク デバイスおよびサービスで使用される、共通のタイプが含まれています。以下の例では、IPv4 アドレスをモデル化しています。

```
typedef ipv4-address {
     type string {
       pattern
         '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-
9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(%[\p{N}\p{L}]+)?';
     }
.
.
```

この例では、一般的な正規表現を使用して IPv4 アドレスをモデル化しています。この正規表現は、有効な IPv4 アドレスをモデル化しています。

ここでは個々の YANG シンタックスについては気にしないでください。YANG ステートメントは新しいものに見えるかもしれませんが、シンタックスの大半にはなじみがあるのではないでしょうか。すでに述べたとおり、`pattern` ステートメントは、ドット付き 10 進表記の一般的な IPv4 アドレスを表した正規表現になります。ここでの目的は、有効な IPv4 アドレスが必要なすべてのデータ モデルで、`ietf-inet-types.yang` データ モデルからこのタイプをいつでも利用できるようにすることです。データ モデルを使用することにより、基本となる API のデータの正確性を確かめることができます。また、一から作り直す必要もなくなります。

YANG データ モデルはどこにあるか

YANG データ モデルは GitHub で確認することができます。YANG リポジトリには多数の共通 YANG データ モデルに加えて、ベンダー固有のモデルも用意されています。YANG データ モデルの例をさらに見るには、このリポジトリを利用する必要があります。

YANGデータ モデルはなぜ重要なのか

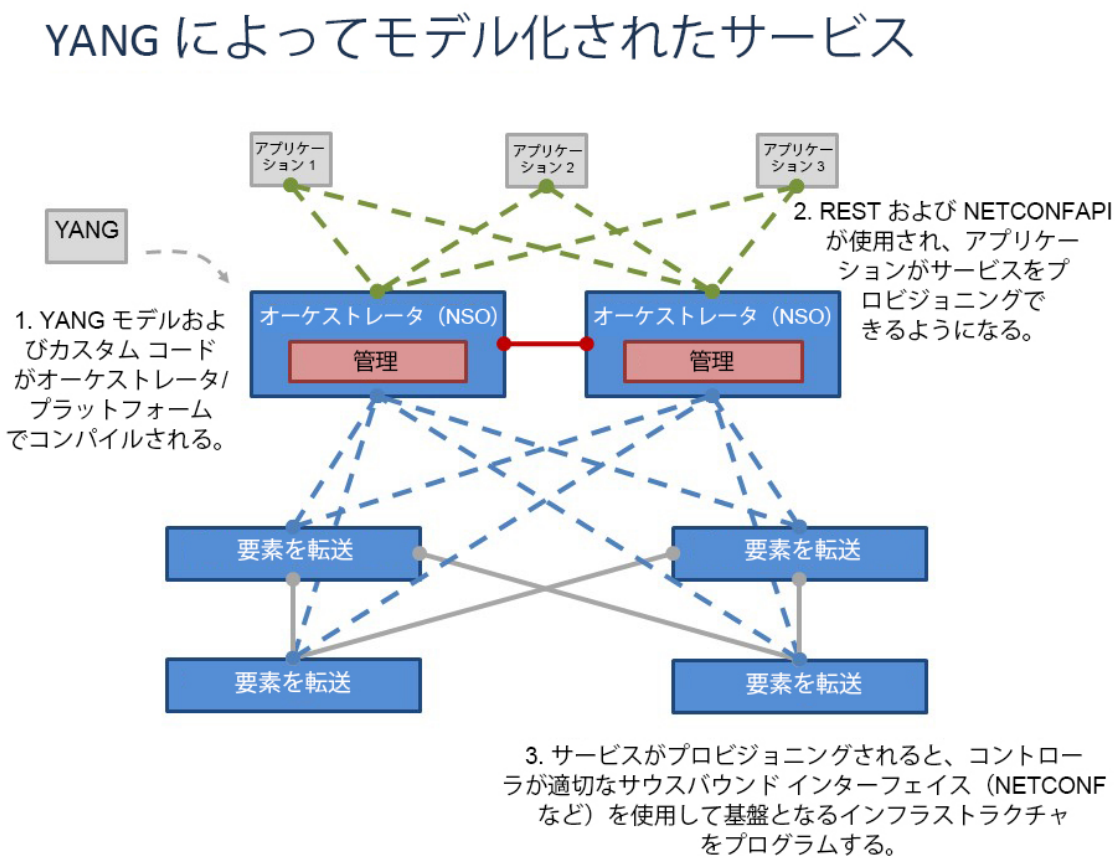
良い質問です。プログラマチック インターフェイスに関する多数の RFC で、YANG モデルは「API コントラクト」と表現されています。私たちも、モジュール 04 で同じ表現を使いました。これは、アプリケーションの作成者がこのデータ モデルに対して開発を行うことで、ベンダーやオペレーティング システムを問わずに API が標準的な方式でサポートされようにできるためです。基本となるデータ モデルを理解すれば、NETCONF や RESTCONF などの API バインディングを使用して、そのデータ モデルに対してプログラミングを行うことができます。

このラボの次のパートでは、このような YANG の使用方法について、さらに詳しく説明します。

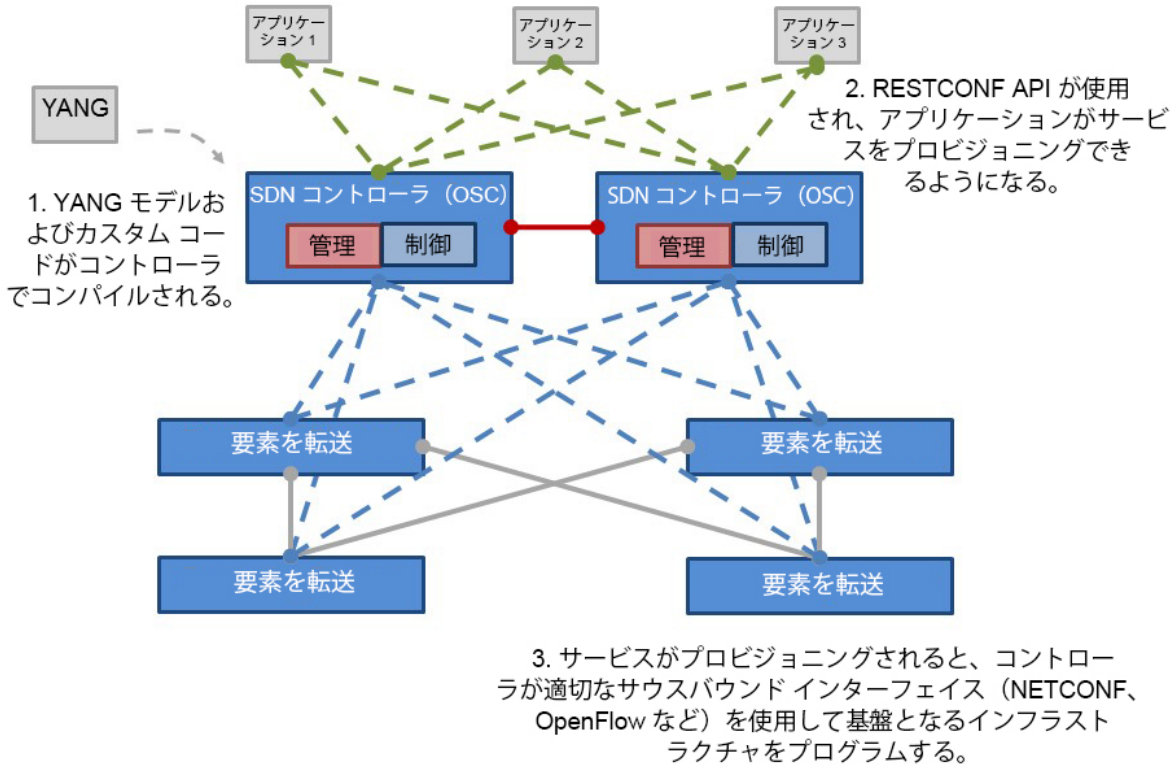
ステップ 3：YANG データ モデルの利用方法を理解する

ネットワーク エンジニアや開発者は YANG データ モデルをさまざまな方法で利用します。YANG の操作について理解するためには、これらの方法を理解することが重要です。

シスコのネットワーク サービス オーケストレータ（NSO）など、一部のオーケストレータでは、YANG データ モデリング言語でネットワーク サービスのモデル化を行うことができます。たとえば、YANG で VPN サービスをモデル化して、このデータ モデルを NSO プラットフォームでコンパイルすることができます。カスタム コードをいくつか作成すると、ユーザによる新規顧客のプロビジョニング時に、NSO が全体のネットワークへ、新しいサービスをオーケストレーションすることができるようになります。下の図は、この概念を説明しています。

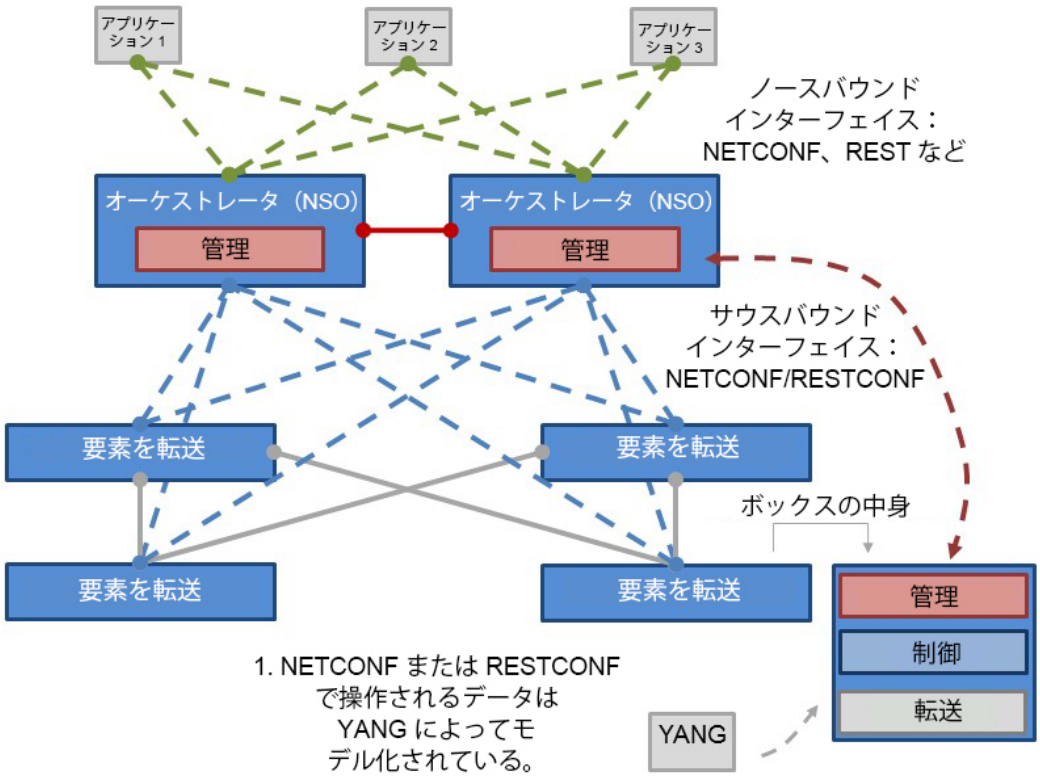


また、Cisco Open SDN Controller（OSC）を使用することにより、アプリケーション（またはサービス）を YANG データ モデリング言語でモデル化することができます。OSC は、OpenDayLight のシスコの製品バージョンです。開発者やエンジニアは、新しいコントロール プレーン アプリケーションや API を YANG でモデル化して、これらを OSC でコンパイルすることができます。さらにカスタム コードを作成して追加すると、OSC を、基盤となるネットワーク用の一元化されたコントロール プレーンまたは管理プレーンとして利用できるようになります。下記の図は、この概念を説明しています。



最後に、個別のネットワーク デバイスでは、YANG を使用してモデル化した NETCONF および RESTCONF インターフェイスが直接使用されます。エンジニアは、OpenConfig などのプロジェクトを使用することで、YANG データ モデリング言語を使用して共通の設定をモデル化します。デバイスで OpenConfig YANG モデルがサポートされていれば、ネットワークのオペレーティング システムやベンダーを問わずに、複数のデバイスで API の一貫性が確保されます。そのため、ネットワーク エンジニアやソフトウェア エンジニアはどちらも、一貫した API を基にアプリケーションやスクリプトを作成することができます。次に示す例では、オーケストレータは、ベンダー製品、シンプルな Python スクリプト、または複雑なカスタム アプリケーションです。

YANG を使用してモデル化された NETCONF/RESTCONF



このラーニング ラボの例では、上記の図に示すように、ネットワーク デバイスでサポートされる YANG データ モデルが使用されます。演習の次のパートでは、実際に YANG を使用して、NETCONF との関係について説明します。

ステップ 4：YANG の実際の動作：実際に試してみる

Cisco CSR1000V の例：NETCONF over SSH を使用して、YANG モデルをサポートしているデバイスに接続する

それでは、YANG の実際の動作を詳しく見ていきましょう。具体的には、まず YANG と NETCONF を接続して、この 2 つの関係を理解できるようにします。

このラボでは、dCloud ラボ環境の CSR1000V を使用します。このデバイスを使用して、YANG によってモデル化されたデータを使用するネットワーク デバイスとやり取りを行う方法について説明します。

始めに、簡単な Python スクリプトと NETCONF を使用して CSR1000V デバイスに接続する、基本的な例を実行します。デバイスへの接続が完了すると、Python スクリプトは CSR1000V（NETCONF サーバ）によってサポートされる NETCONF capabilities を出力します。

```
#!/usr/bin/env python

from ncclient import manager
import sys


# the variables below assume the user is leveraging the
# lab environment and accessing csr1000v
# use the IP address or hostname of your CSR1000V device
HOST = '198.18.133.218'
# use the NETCONF port for your CSR1000V device
PORT = 2022
# use the user credentials for your CSR1000V device
USER = 'admin'
PASS = 'C1sco12345'


# create a main() method
def main():
    """
    Main method that prints netconf capabilities of remote device.
    """
    with manager.connect(host=HOST, port=PORT, username=USER,
                         password=PASS, hostkey_verify=False,
                         device_params={'name': 'default'},
                         look_for_keys=False, allow_agent=False) as m:

        # print all NETCONF capabilities
        print('***Here are the Remote Devices Capabilities***')
        for capability in m.server_capabilities:
            print(capability)

if __name__ == '__main__':
    sys.exit(main())
```

では、上記のコードの動作を見てみましょう。

- まず、ncclient および sys ライブラリをインポートします。
  - *自分のコンピュータで作業している場合は*、このコードを実行する前に、NCClient ライブラリをインストールする必要があります。
  - 最低でも `ncclient` バージョン 0.5.2 を使用しなければならないことに注意してください。
- 次の行は複数の名前を作成します。これらの名前は当該の環境に合わせて必要に応じて更新できます。
  - ラボ環境で CSR1000V を使用している場合は、何も変更する必要はありません。
- 次の行は `main()` メソッドを作成します。このメソッドは、スクリプトがモジュールとしてインポートされた場合ではなく、直接実行された場合にのみ実行されます。
- `with ... as` 式
- ```
  with manager.connect(host=HOST, port=PORT, username=USER,
                       password=PASS, hostkey_verify=False,
                       device_params={'name': 'default'},
                       look_for_keys=False, allow_agent=False) as m:
  ```

  は、必要な引数を使用して NETCONF over SSH セッションを作成します。

  - `host` = リモート デバイスの IP アドレスまたはホスト名。
  - `port` = SSH セッション用の NETCONF ポート。
  - `username` = SSH セッションを認証するためのユーザ名。
  - `password` = SSH セッションを認証するためのパスワード。
  - `hostkey_verify` = ~/.ssh/known_hosts からホストキーを無効にします。
  - `device_params` = ベンダー固有の operations を許可します（この例の場合は NETCONF デバイス）。
  - `look_for_keys` = ユーザ名/パスワードを使用しているため、公開キーを無効にします。
  - `allow_agent` = ユーザ名/パスワードを使用しているため、公開キーを無効にします。
  - `with ... as` 式を使用することにより、ランタイム時に何らかの例外が発生した場合、セッションは正常に終了されます。
- SSH セッションが確立されると、`for` ループが `m.server_capabilities` というリストに対して反復処理されます。このリストには、NETCONF サーバによってアドバタイズされた NETCONF capabilities が含まれています。
- 最後にある `if __name__ == '__main__':` という式は、`main()` メソッドが、スクリプトから直接呼び出された場合にのみ（モジュールとしインポートされた場合ではなく）実行されるようにします。
  ```

セキュリティに関する考慮事項

このラーニング ラボの例では、公開キーを使用するのではなく、ユーザ名/パスワードを Python スクリプトに直接挿入しています。

```
USER = 'admin'
PASS = 'C1sco12345'
```

さらに、このラーニング ラボの例では、`hostkey_verify=False` を使用して、`known_hosts` ファイルを無視しています。

```
with manager.connect(host=HOST, port=PORT, username=USER,
                     password=PASS, hostkey_verify=False,
                     device_params={'name': 'default'},
                     look_for_keys=False, allow_agent=False) as m:
```

実稼働環境では、このような設定を**使用しないでください**。

これらの設定と全体的なアプローチは、ラボのテスト環境の素早い起動と実行に役立っています。

このサンプル コードを実行するには、次のようにします。

1. ステップ 1 で Git リポジトリを複製したディレクトリに移動します。ディレクトリを `devnet-express-code-samples/module06` に変更します。
2. Python スクリプトを実行して、CSR1000V の capabilities を表示します。
    - *Windows の場合*:`py3 get_capabilities.py`
    - *Mac OS または Linux の場合*:`python3 get_capabilities.py`

次のような結果が表示されます。

```
$ python3 get_capabilities.py
***Here are the Remote Devices Capabilities***
urn:ietf:params:xml:ns:yang:smiv2:CISCO-CBP-TC-MIB?module=CISCO-CBP-TC-
MIB&revision=2008-06-24
urn:ietf:params:xml:ns:yang:smiv2:RSVP-MIB?module=RSVP-MIB&revision=1998-08-
25
urn:ietf:params:xml:ns:yang:smiv2:ATM-FORUM-TC-MIB?module=ATM-FORUM-TC-MIB
urn:ietf:params:xml:ns:yang:ietf-packet-fields?module=ietf-packet-
fields&revision=2015-07-14
urn:ietf:params:xml:ns:yang:smiv2:CISCO-ENTITY-EXT-MIB?module=CISCO-ENTITY-
EXT-MIB&revision=2008-11-24
urn:cisco:params:xml:ns:yang:cisco-qos-common?module=cisco-qos-
common&revision=2015-05-09
urn:ietf:params:xml:ns:yang:smiv2:CISCO-NETSYNC-MIB?module=CISCO-NETSYNC-
MIB&revision=2010-10-15
http://tail-f.com/ns/mibs/SNMPv2-MIB/200210160000Z?module=SNMPv2-
MIB&revision=2002-10-16
```

```
urn:cisco:params:xml:ns:yang:cisco-environment?module=cisco-
environment&revision=2015-04-09
http://tail-f.com/ns/mibs/SNMP-NOTIFICATION-MIB/200210140000Z?module=SNMP-
NOTIFICATION-MIB&revision=2002-10-14
urn:ietf:params:xml:ns:yang:smiv2:CISCO-CBP-TARGET-TC-MIB?module=CISCO-CBP-
TARGET-TC-MIB&revision=2006-03-24
urn:ietf:params:netconf:capability:notification:1.0
.
.
.
```

これは capabilities のリストです。とても長いですね。上記の出力は、簡略化のために省略されています。では、この出力の内容をもう少し詳しく見ていきましょう。NETCONF のラーニング ラボで、`hello` のやり取りの間に、NETCONF クライアントとサーバが capabilities をやり取りしたことを思い出してください。YANG を利用して NETCONF/RESTCONF データをモデル化するデバイスの場合、`hello` メッセージをやり取りする間にサポート対象の YANG モジュールの名前がやり取りされます。これは、サポート対象の YANG モジュールを表示するのに、非常に便利な方法です。

それでは、特定の事項に注目するために、この出力の一部を切り出してみましょう。以下のコマンドを使用して、各種の Internet Engineering Task Force（IETF）データ モデルを対象に `grep` 操作を行います。これらのモデルは、各種のプラットフォームに共通して使用されている可能性が高いものです。Windows デバイスと CMD ターミナルを使用している場合は、`grep` コマンドおよび `cut` コマンドの代わりに `findstr` を使用してください。

注：`cut` コマンドは、指定されたデリミタ「?」がある場所で出力行を「カット」します。(`-d?`) を見つけ、モジュール名のみ出力するために、最初のフィールド (`-f1`) のみが出力されます。

```
$ python3 get_capabilities.py | grep ietf- | cut -d? -f1
urn:ietf:params:xml:ns:yang:ietf-packet-fields
urn:ietf:params:xml:ns:yang:ietf-access-control-list
urn:ietf:params:xml:ns:yang:ietf-interfaces
urn:ietf:params:xml:ns:yang:ietf-diffserv-target
urn:ietf:params:xml:ns:yang:ietf-yang-types
urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring
urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing
urn:ietf:params:xml:ns:yang:ietf-netconf-notifications
urn:ietf:params:xml:ns:yang:ietf-interfaces-ext
urn:ietf:params:xml:ns:yang:ietf-yang-smiv2
urn:ietf:params:xml:ns:yang:ietf-diffserv-classifier
urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults
urn:ietf:params:xml:ns:yang:ietf-inet-types
urn:ietf:params:xml:ns:yang:ietf-routing
urn:ietf:params:xml:ns:yang:ietf-diffserv-action
urn:ietf:params:xml:ns:yang:ietf-ospf
urn:ietf:params:xml:ns:yang:ietf-ip
urn:ietf:params:xml:ns:yang:ietf-netconf-acm
urn:ietf:params:xml:ns:yang:ietf-key-chain
urn:ietf:params:xml:ns:yang:ietf-diffserv-policy
urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing
```

いかがでしょう。CSR1000V で、コミュニティ（この場合は IETF）で作成された、いくつかの YANG データ モデルがサポートされていることがわかります。この価値を理解するために、以下の GitHub リポジトリを見てみましょう。これらの YANG モデルは、最新のツール（Git など）を使用して操作されており、さらに大きなコミュニティでオープンに利用したり、検討したりすることができます。それでは、この演習で使用されている YANG の主要な概念をいくつか見てみましょう。

YANG の主要な概念

1. NETCONF のクライアントとサーバは、capabilities をやり取りし、サポート対象の機能を共有します。
    - この例では、`m.server_capabilities` のコンテンツを出力して、CSR1000V デバイスでサポートされている NETCONF capabilities を表示しました。
    - デバイスが YANG データ モデルをサポートしている（NETCONF/RESTCONF をモデリングしている）場合は、capabilities のやり取りの間に YANG モデルの名前がアドバタイズされます。
2. YANG データ モデルは最新のツールを使用して表示できます。
    - YANG モジュールの進展については GitHub で確認することができます。

これで、デバイスが YANG データ モデルをサポートしているかどうかを確認できるようになりました。さて、次の課題です。

次のラボでは、YANG データ モデルの実際の使用方法について説明します。NETCONF を使用し、このデータ モデルに対してプログラミングする方法についても紹介します。

ステップ 5：pyang を使用して YANG データ モデルについて確認する

RESTful Web サービスや各種の API を使用した経験がある人ならば、前の NETCONF に関するラーニング ラボで、API のドキュメンテーションの場所についての説明がなかったのはなぜだろうと思っているかもしれません。その理由の 1 つは、YANG モジュールは「それ自体がドキュメンテーション」であり、API に対してプログラミングを行う方法を理解するのに必要な情報の一部を提供しているからです。YANG データ モジュールは、人とマシン間ではなく、マシン同士の通信を促進するものです。とはいえ、私たちには API を理解するための何らかのツールが必要です。ラーニング ラボのこのステップでは、その方法について説明します。

ここでは、pyang という、オープン ソースの Python パッケージを使用して、この概念を理解します。Pyang は YANG データ モデルを解析し、他のフォーマットに変換して、API の構築方法を理解するのに役立ちます。Pyang は、シンタックス関連の問題について YANG モジュールをテストする場合にも非常に便利です。

それでは、まずいくつかの用語を確認していきましょう。YANG モジュールとは、基本的に YANG のデータ モデリング シンタックスのファイルのことです。YANG モジュールには、以下のような固有のステートメントが含まれています。

- `module`:このステートメントは、YANG モジュールの名前を定義するために使用されます。
- `revision`:このステートメントは、バージョン トラッキングのために使用されます。

YANG モジュールには、モデリング対象のデータを対象とした固有のシンタックスも含まれています。ここで言う YANG モジュールとは、ある固有の YANG ファイルを指していると理解してください。YANG はサブモジュールにモジュール化することができますが、この点については別のラーニング ラボで取り上げます。

前のラーニング ラボでは、サポート対象（具体的には IETF によってサポートされる）のオープン データ モデルの一部について説明しました。それでは、IETF による YANG テータ モデルの操作について詳しく見ていきましょう。まず、dCloud ラボからこの YANG データ モデルを取得するスクリプトの確認から始めます。

```
#!/usr/bin/env python

from ncclient import manager
import sys
import xml.dom.minidom

# the variables below assume the user is leveraging the
# dCloud lab environment and accessing csr1000v
# use the IP address or hostname of your CSR1000V device
HOST = '198.18.133.218'
```

```
# use the NETCONF port for your CSR1000V device
PORT = 2022
# use the user credentials for your CSR1000V device
USER = 'admin'
PASS = 'C1sco12345'
MODULE_NAME = 'ietf-interfaces.yang'


# create a main() method
def main():
    """
    Main method that prints netconf capabilities of remote device.
    """
    with manager.connect(host=HOST, port=PORT, username=USER,
                         password=PASS, hostkey_verify=False,
                         device_params={'name': 'default'},
                         look_for_keys=False, allow_agent=False) as m:

        # print YANG module
        print('***Saving the YANG Module***')
        data = m.get_schema('ietf-interfaces')
        xml_doc = xml.dom.minidom.parseString(data.xml)
        yang_module = xml_doc.getElementsByTagName("data")

        # save the YANG module to a file
        with open(MODULE_NAME, mode='w+') as f:
            f.write(yang_module[0].firstChild.nodeValue)

if __name__ == '__main__':
    sys.exit(main())
```

では、このコードの動作を見てみましょう。

- まず、`ncclient` ライブラリと `sys` ライブラリがインポートされます。
  - 自分のコンピュータで作業している場合は、このコードを実行する前に、NCClient ライブラリをインストールする必要があります。
- `xml.dom.minidom` ライブラリは、XML データを処理する際に役立ちます。
- 次の行は様々な名前を作成します。これらの名前は当該の環境に合わせて必要に応じて更新できます。
  - dCloud ラボ環境で CSR1000V を使用している場合は、何も変更する必要はありません。
- 次の行は `main()` メソッドを定義します。このメソッドは、スクリプトがモジュールとしてインポートされた場合ではなく、直接実行された場合にのみ実行されます。
- `with ... as` 式
- `  with manager.connect(host=HOST, port=PORT, username=USER,`
- `                       password=PASS, hostkey_verify=False,`
- `                       device_params={'name': 'default'},`
- `                       look_for_keys=False, allow_agent=False) as m:`

は、必要な引数を使用して NETCONF over SSH セッションを作成します。

  - `host` = リモート デバイスの IP アドレスまたはホスト名。
  - `port` = SSH セッション用の NETCONF ポート。
  - `username` = SSH セッションを認証するためのユーザ名。
  - `password` = SSH セッションを認証するためのパスワード。
  - `hostkey_verify` = ~/.ssh/known_hosts からホストキーを無効にします。
  - `device_params` = ベンダー固有の operations を有効にします（この例では使用しません）。
  - `look_for_keys` = ユーザ名/パスワードを使用しているため、公開キーを無効にします。
  - `allow_agent` = ユーザ名/パスワードを使用しているため、公開キーを無効にします。
  - `with ... as` 文を使用することにより、ランタイム時に何らかの例外が発生した場合、セッションは正常に終了されます。
- 次に SSH セッションを確立したら、`m.get_schema('ietf-interfaces')` を呼び出して `ietf-interfaces.yang` YANG モデルを取得します。
  - 以前のラーニング ラボで、`hello` プロセスの間にやり取りした capabilities を確認することで、サポート対象の YANG モデルを把握することができました。
- コードの次の行では `<data>` 要素で囲まれた XML ノードが取得されます。結果の文字列は、完全な `ietf-interfaces.yang` データ モデルです。
- 次の行は書き込み対象のファイルを開きます。このファイルは、YANG データ モデルの内容を対象のファイルに書き込むという最後のステップで扱います。
- 最後に、`if __name__ == '__main__':` という文があります。
  - これにより、スクリプトが直接呼び出された場合（モジュールとしてインポートされた場合ではなく）にのみ、`main()` メソッドが実行されるようになります。

このようにスクリプトを実行したら、YANG モデル自体の内容を表示することができます。

このサンプル コードを実行するには、次のようにします。

1. ステップ 1 で Git リポジトリを複製したディレクトリに移動します。ディレクトリを `devnet-express-code-samples/module06` に変更します。
2. 以下のようにして、この Python スクリプトを実行します。
   - *Windows の場合*:`py3 get_ietf_interfaces.py`
   - *Mac OS または Linux の場合*:`python3 get_ietf_interfaces.py`

XML ドキュメントでデバイスから返された YANG データ モデルを確認できます。

```
$ python3 get_ietf_interfaces_yang.py
***Saving the YANG Module***
.
.
.
```

```
$ ls | grep ietf-interfaces.yang
ietf-interfaces.yang
.
.
.
$ more ietf-interfaces.yang
module ietf-interfaces {

  namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";
  prefix if;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     WG Chair: Thomas Nadeau
               <mailto:tnadeau@lucidvision.com>

     WG Chair: Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>";

  description
    "This module contains a collection of YANG definitions for
     managing network interfaces.

     Copyright (c) 2014 IETF Trust and the persons identified as
     authors of the code.All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 7223; see
     the RFC itself for full legal notices.";

  revision 2014/05/08 {
    description
      "Initial revision.";
    reference
      "RFC 7223: A YANG Data Model for Interface Management";
  }
```

```
  /*
   * Typedefs
   */

  typedef interface-ref {
    type leafref {
      path "/if:interfaces/if:interface/if:name";
    }
    description
      "This type is used by data models that need to reference
       configured interfaces.";
  }
  .
  .
  .
```

上記の出力は、YANG データ モデルの内容がファイルに保存されたことを示しています。get-schema NETCONF operation を使用して、ietf-interfaces.yang の YANG データ モデルの内容を取得しました。これで、開発者とマシンが両方とも、デバイスがサポートする YANG データ モデルを理解できるようになります。

ただし、他の方法で YANG データ モデルを取得できることも覚えておいてください。前のラーニング ラボでは、GitHub リポジトリとコミュニティで使用可能な各種の YANG データ モデルについて説明しました。このリポジトリに移動して、ietf-interfaces.yang YANG データ モデルの最新の内容を確認できます。このリポジトリでは、当該の YANG モデルとは少し異なる名前が使用されています。

```
$ git clone https://github.com/YangModels/yang.git
$ more yang/standard/ietf/RFC/ietf-interfaces@2014-05-08.yang
module ietf-interfaces {

  namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";
  prefix if;

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     WG Chair: Thomas Nadeau
               <mailto:tnadeau@lucidvision.com>

     WG Chair: Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>";
```

```
  description
    "This module contains a collection of YANG definitions for
     managing network interfaces.

     Copyright (c) 2014 IETF Trust and the persons identified as
     authors of the code.All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 7223; see
     the RFC itself for full legal notices.";

  revision 2014/05/08 {
    description
      "Initial revision.";
    reference
      "RFC 7223: A YANG Data Model for Interface Management";
  }
  .
  .
  .
```

以上で、使用しているローカル マシンに YANG モデルを保存したので、pyang を使用して内容を確認します。pyang は、YANG データ モデルを解析する Python ツールです。これは、YANG モジュールの確認に非常に便利なツールです。

注:CSR1000V から取得した、ietf-interfaces.yang ファイルのローカル コピーがあるディレクトリにいることを確認してください。

```
$ pyang ietf-interfaces.yang
```

さて、意外な結果になりました。この方法で pyang を実行しても、何も出力されません。何が起こっているのかを把握するために、YANG モジュールにおけるシンタックスの問題を確認します(vi/vim に慣れていなければ、お好みのエディタを使用してください)。

```
$ # delete the 'm' in the 'module' statement in the first line
$ vim ietf-interfaces.yang
odule ietf-interfaces {
.
.
.
```

上記の変更を保存し、pyang を実行して出力を確認してください。

```
$ pyang ietf-interfaces.yang
ietf-interfaces.yang:1: error: unexpected keyword "odule", expected one of
['module', 'submodule']
$
```

さて、pyang で YANG モデルを解析する理由はさまざまです。その 1 つが、上記の YANG シンタックスの問題を確認することです。それでは、前のステップで紹介したシンタックスの問題を修正しましょう。

```
$ # fix the syntax issue by adding the 'm' back in the 'module' statement
$ vim ietf-interfaces.yang
module ietf-interfaces {
.
.
.
```

必要に応じて pyang を再度実行し、YANG モジュールが修正されたことを確認してください。

先に進む前に、このラボで取り上げた、YANG の重要な項目について再度確認してください。

YANG の主要な概念

1. YANG モデルの確認
    - このステップの例では、YANG 内部の仕組みについて確認しました。
        - GitHub またはデバイスから YANG モデルを取得できることを確認しました。
2. YANG はモジュールで構成されている
    - モジュールとは、YANG データ モデル シンタックスのファイルです。

ラボの次のステップでは、ietf-interfaces.yang モジュールを詳しく検討することによって、YANG についてさらに詳しく理解します。

ステップ 6：YANG の基本に関する追加情報および pyang の操作

ラーニング ラボの次のステップでは、YANG の基本に関する追加情報と、NETCONF との関係について確認します。また、pyang を使用して、基盤となる YANG データ モデルを理解する方法を理解します。

pyang を使用して YANG データ モデルを解析する

それでは、再度 pyang にアクセスして、YANG データ モデル全体を解析しましょう。pyang は、YANG データ モデルを検証し、他のフォーマットに変換するのに非常に便利なツールです。以下の pyang の実行例を見てみます。pyang に慣れる必要がある場合は、これをご使用のホストまたは Windows VM で実行できます。

```
$ pyang -f tree ietf-interfaces.yang
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                          string
   |     +--rw description?                  string
   |     +--rw type                          identityref
   |     +--rw enabled?                      boolean
   |     +--rw link-up-down-trap-enable?     enumeration {if-mib}?
   +--ro interfaces-state
      +--ro interface* [name]
         +--ro name               string
         +--ro type               identityref
         +--ro admin-status       enumeration {if-mib}?
         +--ro oper-status        enumeration
         +--ro last-change?       yang:date-and-time
         +--ro if-index           int32 {if-mib}?
         +--ro phys-address?      yang:phys-address
         +--ro higher-layer-if*   interface-state-ref
         +--ro lower-layer-if*    interface-state-ref
         +--ro speed?             yang:gauge64
         +--ro statistics
            +--ro discontinuity-time    yang:date-and-time
            +--ro in-octets?            yang:counter64
            +--ro in-unicast-pkts?      yang:counter64
            +--ro in-broadcast-pkts?    yang:counter64
            +--ro in-multicast-pkts?    yang:counter64
            +--ro in-discards?          yang:counter32
            +--ro in-errors?            yang:counter32
            +--ro in-unknown-protos?    yang:counter32
            +--ro out-octets?           yang:counter64
            +--ro out-unicast-pkts?     yang:counter64
            +--ro out-broadcast-pkts?   yang:counter64
            +--ro out-multicast-pkts?   yang:counter64
            +--ro out-discards?         yang:counter32
            +--ro out-errors?           yang:counter32
```

この例は、YANG データ モデルを解析して、基になっているデータの基盤となるツリー構造について理解する方法を示しています。このツリー出力では、読み取り/書き込み（rw と読み取りのみ（ro）のデータを明確に区別できます。各種のデータのうち、rw というラベルが付いたものは設定データであり、ro というラベルが付いたものは operational データです。これは、NETCONF プロトコルでは、設定データと operational データの文字列の分離が強制されるためです。参考までに、この分離を強制するために、YANG では config ステートメントが使用されます。また、データをツリー形式で表示する理由の 1 つは、この形式がネットワーク設定の構造化方法に対応していることです。ネットワーク設定は一般にツリーで構造化されているため、YANG データ モデルにおいてもデータを同様の形式でモデル化しています。

条件ステートメントおよびフロー制御ステートメントの場所

ここまでに、YANG に組み込まれている重要なシンタックスとデータ タイプをいくつか取り上げてきました。フロー制御については取り上げられるのか、気になっている方がいるかもしれません。

ここで、YANG がデータ モデリング言語であることを思い出してください。YANG モジュールは Python スクリプトや Perl スクリプトのように実行されるものではありません。YANG は NETCONF や RESTCONF がアクセスするデータをモデル化するために使用されます。YANG モジュールは、confD など Cisco's Open SDN Controller、アプリケーションやプラットフォームによってコンパイルされ、プログラマチック インターフェイスとして使用されます。

この概念を理解するために、別の例を見てみましょう。あなたはデータ モデラーであり、サーバまたはネットワーク デバイス上のインターフェイスを概念的にモデル化しようとしているとします。外部に対して公開する必要があるのはどのデータ プロパティでしょうか。以下のようなものが考えられます。

- インターフェイス カウンタ：処理されたパケットやバイトの数
- Operational 状態：インターフェイスは稼働中か、ダウンしているか。
- 物理アドレス：インターフェイスの物理アドレス（イーサネットの MAC アドレスなど）、接続状態をトラブルシューティングすることができます。

これらは、データ モデラーとして必要になる質問の例です。YANG データ モデリング言語には、このようなデータ プロパティをモデル化するためのシンタックスが用意されています。たとえば、物理アドレス（MAC アドレスなど）に対するクエリが 1 つの整数を返すことがあるでしょうか。ありえません。YANG を使用することで、適切なデータ シンタックスを適用することができます。

ここでは、YANG の興味深い概念のごく一部を紹介しました。次のラーニング ラボを続ける前に、これらの概念について復習しておきましょう。

YANG の主要な概念

1. YANG はデータ モデリング言語（プログラミング言語ではない）
    - スクリプトまたはアプリケーションを作成する場合は、フロー制御メカニズムを使用して、ロジックのフロー全体を検討する必要があります。
    - YANG は、データのシンタックスやセマンティクスの適切な適用を実現するデータ モデリング言語にすぎません。
        - 例:
            - ユーザに対して、ある API コール（例：00:aa:bb:cc:11:22:33）から物理アドレスが返されるようにするとします。
            - データが正しくフォーマットされ、プログラマーが予期される内容を理解できるようにするために、YANG の `pattern` ステートメントの後に正規表現を付けて使用することができます。
2. `pyang` は YANG データ モデル全体の解析に欠かせないツールです。
    - その他の例としては、YANG explorer などがあります。
    - `pyang` を使用することで YANG データ モデル全体を解析し、基盤となるデータについて理解することができます。
    - 非常に重要なことですが、YANG データ モデルには「API ドキュメンテーション」としての役割があります。

とても便利ですね。ラボの次のステップでは、YANG についてのこの知識を応用して、YANG データ モデルに基づいた NETCONF API コールを行う方法について学びます。

ステップ 7：YANG データ モデルから API を実行するためのロジック

ここまで、YANG データ モデルに関する多くの情報を取り上げてきました。それでは、こうした知識を応用して NETCONF API コールを行う方法を見てみましょう。

ラボの最後のステップでは、YANG データ モデルを利用して、NETCONF を使用するデバイスに対してプログラミングを行う方法を紹介します。dCloud ラボ環境内の CSR1000V から、インターフェイスの設定をプログラムで取得する必要があるとします。このタスクを、一般的な YANG モデルである `ietf-interfaces.yang` を使って実行する方法を見てみましょう。

YANG データ モデルから NETCONF の API コールを作成する

まず、これまで学んできた NETCONF API に関する知識を一般的なケースに応用する方法についてのロジックを見ていきます。次のリストは、NETCONF を使用する YANG データ モデルに対してプログラミングを行う方法を決定するための手順を示すものです。具体的には、`ietf-interfaces.yang` に対するプログラミングについて、このロジックを使用します。

前のラーニング ラボの手順で学んだロジックを思い出してください。このラボでは、API ロジックを作成するために次のようなタスクを実行します。

1. サポート対象の YANG モジュールを検証する
    - ラーニング ラボでは、`get_capabilities.py` スクリプトを使用して、アプローチする方法を学びました。
    - 最初のステップは、アドバタイズされている capabilities のリストから返されるモデルを表示することで、必要な YANG データ モデルがデバイスでサポートされているかどうかを確認することです。
2. YANG モジュールを解析する
    - 次のステップは、YANG データ モデルを解析して、適切な API の構造を把握することです。
    - これらのラーニング ラボでは、`pyang` を使用して、YANG モジュールと関連付けられた API を確認しています。
3. NETCONF のロジックを適用する
    - RPC operation を決定する必要があります。
    - 設定を取得するには、`get-configuration` operation を使用する必要があります。
4. 適切な XML を構築する
    - RPC operations で XML ドキュメントを使用して、特定のデータをフィルタ処理したり、設定したりできます。
    - 適切なフィルタを作る必要があります。このフィルタは YANG データ モデルから決定できます。

5. XML の応答の解析
    - XML データを取得した場合は、各 Python モジュールのメソッドを使用して、対応する応答を解析する必要があります。

次の一連のタスクでは、上記の各ステップを確認して、YANG でモデル化された NETCONF API に対するプログラミングを行う方法を説明します。

実際に試してみる

上記の手順を実行して、ニーズに合った NETCONF API コールを作成しましょう。

1. サポート対象の YANG モジュールを確認します。

   サポートされる YANG モジュールのリストを取得するためには、基本的な Python スクリプトを使用しました。このリストは `hello` メッセージとともに交換されました。以下に示すように、`ietf-interfaces.yang` がサポートされています。

```
 $ python get_capabilities.py | grep ietf-interfaces
 urn:ietf:params:xml:ns:yang:ietf-interfaces?module=ietf-
interfaces&revision=2014-05-08&features=pre-provisioning,if-
mib,arbitrary-names&deviations=ietf-ip-devs
```

   とても便利ですね。再度、capabilities のやり取りから、YANG データ モデルが実際にデバイスでサポートされていることを確認してみましょう。

2. YANG モジュールを解析します。

   前のラボの手順では、`pyang` を使用して YANG モデルを解析しました。以下の出力は、該当する部分を示すため、省略されています。この出力からは、この YANG モジュールの NETCONF XML の解析方法を推測することができます。

```
pyang -f tree ietf-interfaces.yang
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                          string
   |     +--rw description?                  string
   |     +--rw type                          identityref
   |     +--rw enabled?                      boolean
   |     +--rw link-up-down-trap-enable?     enumeration {if-mib}?
   .
   .
   .
```

結果の階層データを見てください。YANG は XML を使用する NETCONF をモデル化して、ネットワーク設定用にデータをシリアライズすることを目的としているため、このインデント処理は意図的なものです。上記から、この後の手順で示すとおり、XML が必要であることが推測されます。

3. NETCONF のロジックを適用します。

   NETCONF を使用して設定を取得するには、一般には `get-config` operation を使用します。

   また、NETCONF サーバのデータストアでは設定データと operational データが分離されているため、NETCONF サーバに対して、どこからデータを取得するのかを指示する必要もありました。この場合は、running configuration からデータを取得します。

4. 適切な XML を構築します。

   `get-config` RPC を発行する場合は、必要な設定の該当部分をフィルタ処理する必要があります。この処理を行わないと、デバイスの設定がすべて取得されます。このシナリオでは、必要なのはインターフェイスの設定情報のみです。この `pyang` の出力をガイドとして使用することで、該当する XML ドキュメントは以下のようになると想定されます。

   ```
   <filter>
     <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
       <interface></interface>
     </interfaces>
   </filter>
   ```

   上記の XML は、この後すぐに作成するスクリプトの `get_interfaces.xml` の内容です。

   XML ドキュメントの実際の構造は、YANG データ モデルで使用されるタイプによって決まります。ただし、`pyang` を使用すれば、前のツリー階層にあるものと類似した XML ドキュメントを作成できます。

   > **注:** 上記のフィルタでは、目的の YANG データ モデルを指す適切な名前空間が必要です。このフィルタを参照することで、該当するインターフェイスの設定を取得できます。

例の確認

それでは、目的の設定データを取得するための、Python スクリプトの例を確認しましょう。

```
#!/usr/bin/env python

from ncclient import manager
import sys
```

```
import xml.dom.minidom


# the variables below assume the user is leveraging the
# dCloud lab environment and accessing csr1000v
# use the IP address or hostname of your CSR1000V device
HOST = '198.18.133.218'
# use the NETCONF port for your CSR1000V device
PORT = 2022
# use the user credentials for your CSR1000V device
USER = 'admin'
PASS = 'C1sco12345'
# XML file to open
FILE = 'get_interfaces.xml'

# create a main() method
def get_configured_interfaces(xml_filter):
    """
    Main method that retrieves the interfaces from config via NETCONF.
    """
    with manager.connect(host=HOST, port=PORT, username=USER,
                         password=PASS, hostkey_verify=False,
                         device_params={'name': 'default'},
                         allow_agent=False, look_for_keys=False) as m:
        with open(xml_filter) as f:
            return(m.get_config('running', f.read()))


def main():
    """
    Simple main method calling our function.
    """
    interfaces = get_configured_interfaces(FILE)
    print(xml.dom.minidom.parseString(interfaces.xml).toprettyxml())

if __name__ == '__main__':
    sys.exit(main())
```

では、このコードの動作を見てみましょう。

- まず、ncclient および sys ライブラリをインポートします。
  - 自分のコンピュータで作業している場合は、このコードを実行する前に、NCClient ライブラリをインストールする必要があります。
- 次の行は複数の名前を作成します。これらの名前は当該の環境に合わせて必要に応じて更新できます。
  - dCloud ラボ環境で CSR1000V を使用している場合は、何も変更する必要はありません。
- 次の行では、`get_configured_interfaces()` という関数が作成されます。この関数は、`main()` メソッドで使用されます。スクリプトの数行下で、この `main()` メソッドが呼び出されます。

- コードの次の行では、`get_configured_interfaces()` 関数で以下の `connect` メソッドが使用されています。
- `  with manager.connect(host=HOST, port=PORT, username=USER,`
- `                       password=PASS, hostkey_verify=False,`
- `                       device_params={'name': 'default'},`
- `                       allow_agent=False, look_for_keys=False) as m:`
- このようにして接続を作成することで、以下の引数を使用して NETCONF over SSH セッションが作成されます。
    - `host` = リモート デバイスの IP アドレスまたはホスト名。
    - `port` = SSH セッション用の NETCONF ポート。
    - `username` = SSH セッションを認証するためのユーザ名。
    - `password` = SSH セッションを認証するためのパスワード。
    - `hostkey_verify` = ~/.ssh/known_hosts からホストキーを無効にします。
    - `device_params` = ベンダー固有の operations を有効にします（この例ではデフォルト）。
    - `look_for_keys` = ユーザ名/パスワードを使用しているため、公開キーを無効にします。
    - `allow_agent` = ユーザ名/パスワードを使用しているため、公開キーを無効にします。
    - `with/as` 文を使用することにより、ランタイム時に何らかの例外が発生した場合、セッションは正常に終了されます。
- SSH セッションを確立したら、次に `m.get_config('running', f.read())` を呼び出します。
    - この例では、`f` は `get_interfaces.xml` という XML ファイルを参照します。このファイルには、XML フィルタについて説明したステップで取り上げた XML フィルタが含まれています。
    - NETCONF operation の結果は `return` ステートメントを使用して返されます。
- コードの次の行では、`main()` メソッドが作成されます。このメソッドは、`ietf-interfaces.yang` によってモデル化された、該当する設定を取得するための関数を呼び出します。
- 最後に、`if __name__ == '__main__':` という文があります。
    - これにより、スクリプトが直接呼び出された場合（モジュールとしてインポートされた場合ではなく）にのみ、`main()` メソッドが実行されるようになります。

このスクリプトを以下のように実行すると、結果としてデバイスのインターフェイスの設定が表示されます。

このサンプル コードを実行するには、次のようにします。

1. ステップ 1 で Git リポジトリを複製したディレクトリに移動します。ディレクトリを `devnet-express-code-samples/module06` に変更します。

2. 以下の Python コードを実行します。
    - *Windows の場合*:`py3 get_interfaces_config.py`
    - *Mac OS または Linux の場合*:`python3 get_interfaces_config.py`

インターフェイスの設定が、XML フォーマットで表示されます。

```
$ python3 get_interfaces_config.py
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:e7570e5c-5dd9-11e6-8001-3c15c2bc7ae8"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
            <interface>
                <name>GigabitEthernet1</name>
                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-
type">ianaift:ethernetCsmacd</type>
                <enabled>true</enabled>
                <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
                    <address>
                        <ip>198.18.133.212</ip>
                        <netmask>255.255.192.0</netmask>
                    </address>
                </ipv4>
                <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
            </interface>
        </interfaces>
    </data>
</rpc-reply>
$
```

これで、YANG に基づいた NETCONF API コールを生成するプロセスの概要を把握することができました。pyang の出力に比べれば、XML にはなじみがあるのではないでしょうか。下記の図は、Yang データ モデルからの XML の作成と、YANG ステートメントと実際の API との対応を示しています。

次のラーニング ラボでは、同様のアプローチにより、RESTCONF プロトコルと関連する API についてさらに詳しく学びます。この調子で頑張ってください。