デバイスレベルインターフェイス: NETCONF と YANG

Tetsuhiro Sato

DevNet Express
DNA Programmability

cisco DevNet
developer.cisco.com

DevNet → Discover → Learning Tracks → DNA → Module LM-4602

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

Beta

LM-4602: Device Level Interfaces: NETCONF / YANG

RESTCONF, NETCONF / YANG と、Python について学びます。

2 Hours 10 Minutes

NETCONF RESTCONF EXPRESS

Introduction to Standard Device Interfaces (Japanese)
NETCONF/YANG がネットワーク管理テクノロジーにどのように適合するかを理解します。

Introduction to YANG Data Modeling (Japanese)
データ モデルとは何か、そしてネットワーク管理における YANG の役割が何かについて学びます。

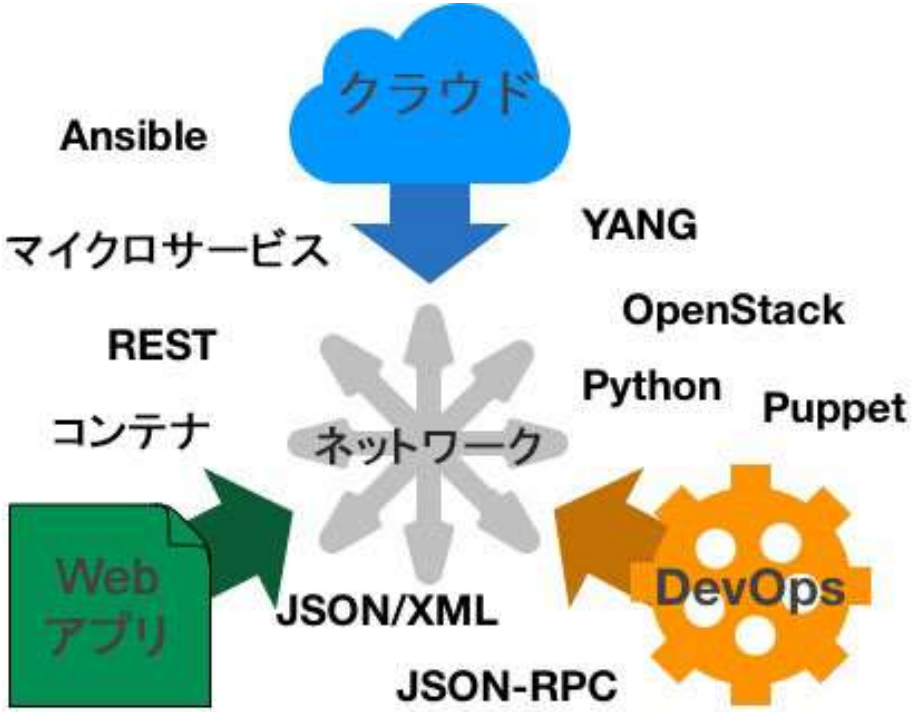
Introduction to the NETCONF Protocol (Japanese)
NETCONF プロトコルの重要な要素と使用方法の確認

Mission: Programming with NETCONF and Python (Japanese)
NETCONF/YANG を使用してインターフェイスの統計を取得して Spark にポストするための Python スクリプトを作成する

Login to Start Module

標準デバイスインターフェイス
イントロダクション

ネットワークはもはや独立して動作しない



SNMPとは何だったのか?

- 主にデバイス監視に利用: SNMPv2 リードオンリーMIB
- 典型的な利用方法: インターフェイス統計へのクエリとトラップ
- 経験的な観測: SNMPはコンフィグ用途には向かない
  - 書き込み可能なMIBが少ない
  - セキュリティ的な懸念
  - リプレイ / ロールバックが難しい
  - 特化したアプリケーション向き

SNMP works "reasonably well for device monitoring"

SNMPってデバイス監視にはいいよね

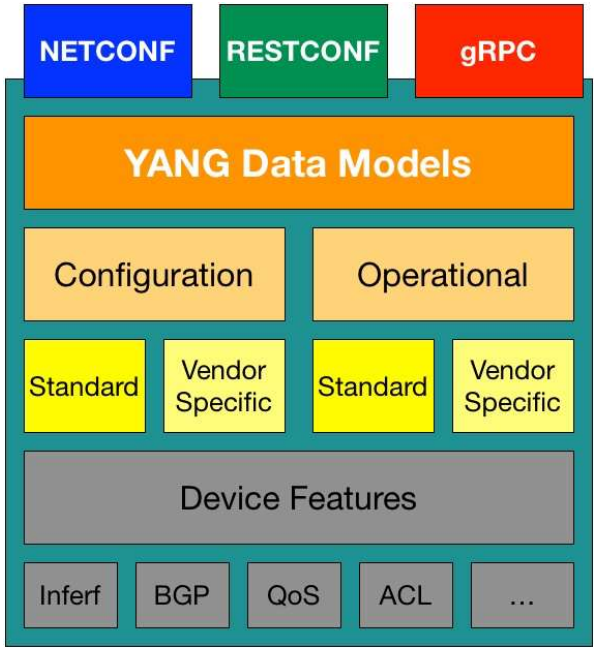
RFC 3535: Overview of the 2002 IAB Network Management Workshop – 2003
https://tools.ietf.org/html/rfc3535

RFC 3535: では何が必要?

- デバイスコンフィグのためのプログラミング可能なインターフェイス
- 構成(コンフィグ)と状態(ステート)のデータの分離
- デバイスではなくサービスに対してコンフィグできること
- エラーチェックとリカバリが組み込まれていること

NETCONF / YANG (RESTCONF & gRPC)

- NETCONF – 2006 – RFC 4741
  (RFC 6241 in 2011)
- YANG – 2010 – RFC 6020
- RESTCONF – 2017 – RFC 8040
- gRPC – 2015 – オープンソースプロジェクト by Google



NETCONFにおけるプロトコルとデータモデルの

- NETCONFは**YANGデータモデルが定義される数年前に**トランスポートプロトコルとして標準化
- YANGモデルが存在する前の時点で、NETCONFは**ベンダー固有のデータモデルを利用**
- 今後はYANGはネットワークにおける主要なデータモデルになりえるが、**YANGデータではないデータを提供するデバイスも引き続き存在する可能性もある**
- YANGデータモデルはNETCONFとは別に活用することができる。**RESTCONF** がよい例

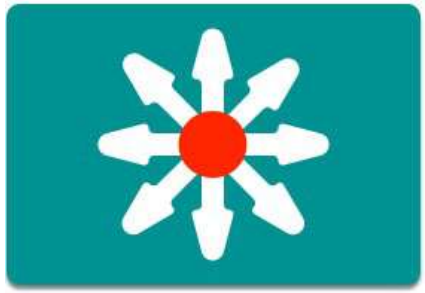
データモデルとは?

データモデルは、単に「何か」を記述するために広く理解され、合意された方法です

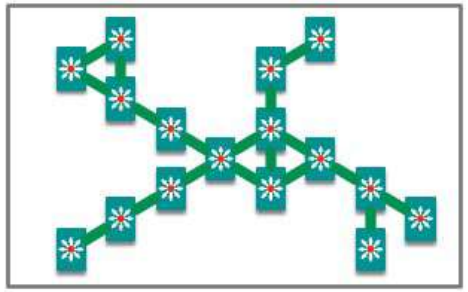
例として、人のためのこの単純な「データモデル」を考えてみましょう

- 人
  - **性別** - 男、女、その他
  - **身長** - 何センチ?
  - **体重** - 何キロ?
  - **髪の色** - ブラウン、ブロンド、黒、赤、その他
  - **目の色** - ブラウン、ブルー、グリーン、栗色、その他

YANG データモデルは何を表現するのか?



デバイス データ モデル
- インターフェイス
- VLAN
- デバイス ACL
- トンネル
- OSPF
- その他



サービス データ モデル
- L3 MPLS VPN
- MP-BGP
- VRF
- ネットワーク ACL
- システム管理
- ネットワーク障害
- その他

データモデル vs データフォーマット

Text

```
switch# show int bri

--------------------------------------------------------------------------------
Ethernet      VLAN    Type Mode   Status  Reason                   Speed     Port
Interface                                                                    Ch #
--------------------------------------------------------------------------------
Eth1/1        1       eth  fabric down    SFP not inserted           10G(D) --
```

JSON

```
{
    "interfaces": [
        {
            "interface": "Ethernet1/1",
            "vlan": "1",
            "type": "eth",
            "portmode": "fabric",
            "state": "down",
            "state_rsn_desc": "SFP not inserted",
            "speed": "10G",
            "ratemode": "D"
        }
    ]
}
```

同じデータを
異なるフォーマットで
表現できる

XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:ba
 <nf:data>
  <show>
   <interface>
    <__XML__INTF_ifeth_brf>
     <__XML__PARAM_value>
      <__XML__INTF_output>Ethernet1/1-3</__XML__INTF_outp
     </__XML__PARAM_value>
     <__XML__OPT_Cmd_show_interface_if_eth_brief___readon
      <__readonly__>
       <TABLE_interface>
        <ROW_interface>
         <interface>Ethernet1/1</interface>
         <vlan>1</vlan>
         <type>eth</type>
         <portmode>fabric</portmode>
         <state>down</state>
         <state_rsn_desc>SFP not inserted</state_rsn_desc
         <speed>10G</speed>
         <ratemode>D</ratemode>
        </ROW_interface>
       </TABLE_interface>
      </__readonly__>
     </__XML__OPT_Cmd_show_interface_if_eth_brief___reado
    </__XML__INTF_ifeth_brf>
   </interface>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

YANG データモデル
イントロダクション

“YANG” の3つの意味

- YANG モデリング言語
- YANG データモデル
- YANG デバイスデータ

モデルの種類

業界標準

- **標準団体による定義**
  (IETF, ITU, OpenConfig, etc.)
- **標準に準拠**
  `ietf-diffserv-policy.yang`
  `ietf-diffserv-classifer.yang`
  `ietf-diffserv-target.yang`

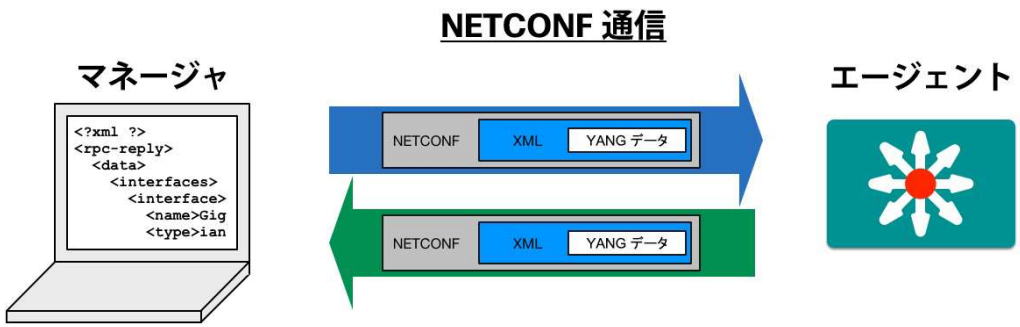
ベンダ独自

- **ベンダによる定義**
  (i.e. Cisco)
- **ベンダプラットフォーム固有**
  `cisco-memory-stats.yang`
  `cisco-flow-monitor`
  `cisco-qos-action-qlimit-cfg`

https://github.com/YangModels/yang

NETCONF
イントロダクション

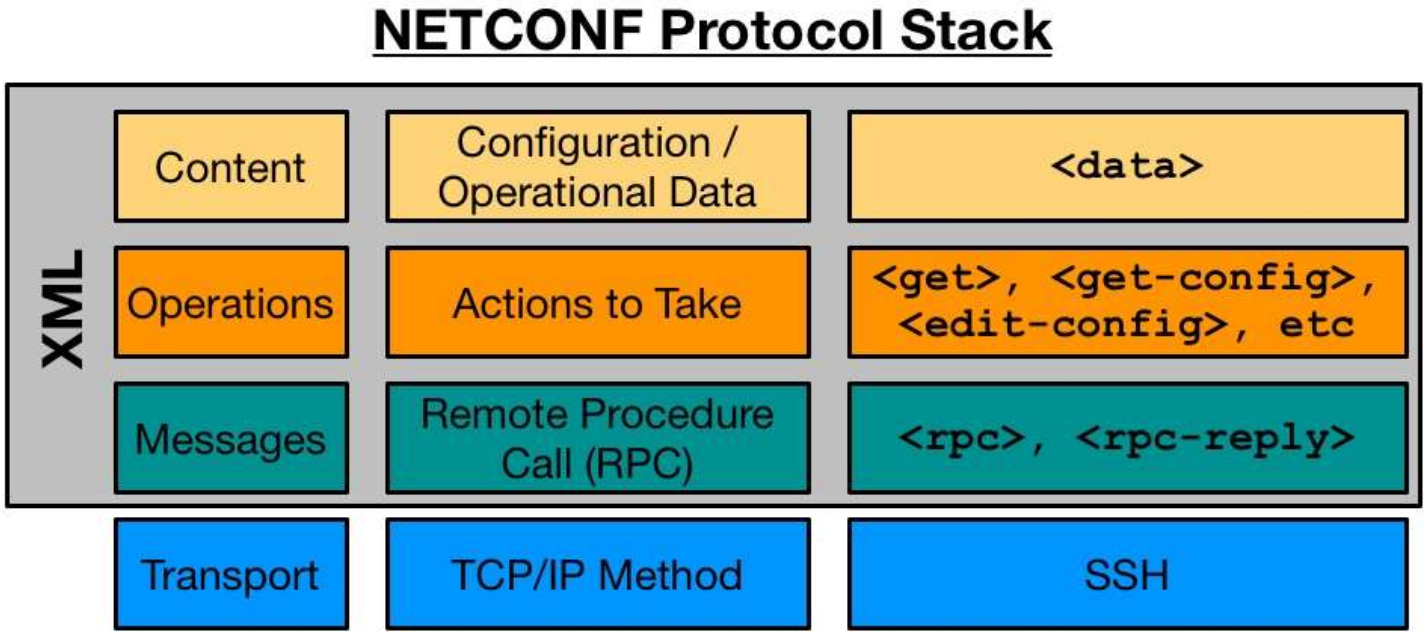
NETCONF プロトコルのイントロダクション



Some key details:

- 2006 年に RFC4741 で最初の標準化
- 2011年に RFC6241 で最新の標準化
- コンテンツは明示的には定義しない

NETCONF プロトコルスタック



NETCONF レビュー

Transport - SSH

```
$ ssh admin@192.168.0.1 -p 830 -s netconf
admin@192.168.0.1's password:
```

SSH Login

```
<hello xmlns="urn:ietf:params:xml:
<capabilities>
  <capability>urn:ietf:params                    e:1.1</cap
  <capability>urn:ietf:param                     bility:candi                    /capability>
  <capability>urn:ietf:para                      f-mo                    </capability>
  <capability>urn:ietf:par                       ns                    interfaces<                    ity>
  [output omitted and edit                       larif
</capabilities>
<session-id>19150</session                       ello>]]>]]>
```

Server (Agent) sends hello

```
<?xml version="1.0" encodin                      ">
<hello xmlns="urn:ietf:param                     etconf:base:
<capabilities>
  <capability>urn:ietf:params:ne
</capabilities>
  </hello>]]>]]>
```

Client (Manager) sends hello

Don't NETCONF like this

Example edited for simplicity and brevity

NETCONF と Python: ncclient

```
from ncclient import manager

m = manager.connect(host="192.168.0.1",
                    port=830,
                    username="admin",
                    password="cisco123",
                    hostkey_verify=False
                    )

m.close_session()
```

Full Info at: http://ncclient.readthedocs.io/en/latest/

オペレーション - NETCONF アクション

Operation	Description
<get>	実行中のコンフィギュレーションおよびデバイスの状態情報を取得する
<get-config>	指定されたコンフィギュレーション データストアの全体または一部を取得する
<edit-config>	指定されたコンフィギュレーション データストアにすべての設定のまたは一部の設定をロードする
<copy-config>	コンフィギュレーション データストア全体を他で置き換える
<delete-config>	コンフィギュレーション データストアを削除する
<commit>	実行中のデータストアに候補のデータストアをコピーする
<lock> / <unlock>	コンフィギュレーション データストア システム全体をロックまたはロック解除する
<close-session>	NETCONF セッションの終了
<kill-session>	NETCONF セッションの強制終了

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

Lab #2:

YANG データ モデルのイントロダクション

- Step 3 – YANG データ モデル
  https://learninglabs.cisco.com/tracks/devnet-express-dna-jp/06-dmi-jp/06-dmi-02-introducing-yang-data-modeling-jp/step/3

pyang による出力

```
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
           [OUTPUT REMOVED]
```

Module Name

Key

Leaf

Type

container

list

container

list

分かりやすく単純化された例

DevNet → Discover → Learning Tracks → DNA → Module LM-4602

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

Lab #2:

YANG データ モデルのイントロダクション

- Step 4 – YANG でモデル化された実際のデバイス データ
  https://learninglabs.cisco.com/tracks/devnet-express-dna-jp/06-dmi-jp/06-dmi-02-introducing-yang-data-modeling-jp/step/4

実際のデバイスデータの出力

interfaces container

interface node

```
<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>
  <interface>
      <name>GigabitEthernet3</name>
      <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
      <enabled>false</enabled>
      <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
      <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
  </interface>
</interfaces>
```

Namespace = Capability = Model

Leaf

分かりやすく単純化された例

DevNet → Discover → Learning Tracks → DNA → Module LM-4602

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

Lab #3:

Introduction to NETCONF

- Step 3 – NETCONF 通信に ncclient を使用
  https://learninglabs.cisco.com/tracks/devnet-express-dna-jp/06-dmi-jp/06-dmi-03-introducing-the-netconf-protocol-jp/step/3

コードレビュー – Part 1

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

```
from ncclient import manager
import sys
```

ライブラリのインポート

```
# the variables below assume the user is leveraging the
# network programmability lab 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'
```

変数のセット

コードレビュー – Part 2

```
# 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.split('?')[0])

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

接続してhelloの交換

ケイパビリティをプリント

DevNet → Discover → Learning Tracks → DNA → Module LM-4602

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

Lab #3:

NETCONFのイントロダクション

- Step 5 – デバイスのホスト名を取得(例3)
  https://learninglabs.cisco.com/tracks/devnet-express-dna-jp/06-dmi-jp/06-dmi-03-introducing-the-netconf-protocol-jp/step/5

コードレビュー (Just the new parts…)

```
import xml.dom.minidom
```

XML ライブラリのインポート

```
# create a main() method
def main():
        [SECTION REMOVED]
        # XML filter to issue with the get operation
        hostname_filter = '''
                          <filter>
                              <native xmlns="urn:ios">
                                  <hostname></hostname>
                              </native>
                          </filter>
                          '''
        result = m.get_config('running', hostname_filter)
        xml_doc = xml.dom.minidom.parseString(result.xml)
        hostname = xml_doc.getElementsByTagName("hostname")
        print(hostname[0].firstChild.nodeValue)
```

XML フィルタで取得する情報を特定

<get-config>

XML 処理で “hostname” を取得

分かりやすく単純化された例

DevNet → Discover → Learning Tracks → DNA → Module LM-4602

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

Lab #3:

NETCONFのイントロダクション

- Step 7 – IETF インターフェイスのクエリ(例4)
  https://learninglabs.cisco.com/tracks/devnet-express-dna-jp/06-dmi-jp/06-dmi-03-introducing-the-netconf-protocol-jp/step/7

XML Output レビュー

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:823b5318-248a-4247-aa74-c94eda2fd9cf" 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>
      <interface>
        <name>GigabitEthernet3</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>false</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

Identify output as XML

Namespace = Capability = Model

Leaf

rpc message

data block

interfaces container

interface node

分かりやすく単純化された例

コードレビュー (Just the new parts…)

```
# XML file to open
FILE = 'get_interfaces.xml'
```

XMLフィルタに使うファイル名

```
# 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()))
```

ファイルを開いて<get-config>の際のXMLフィルタとして利用

Example edited for simplicity and brevity

ご参考

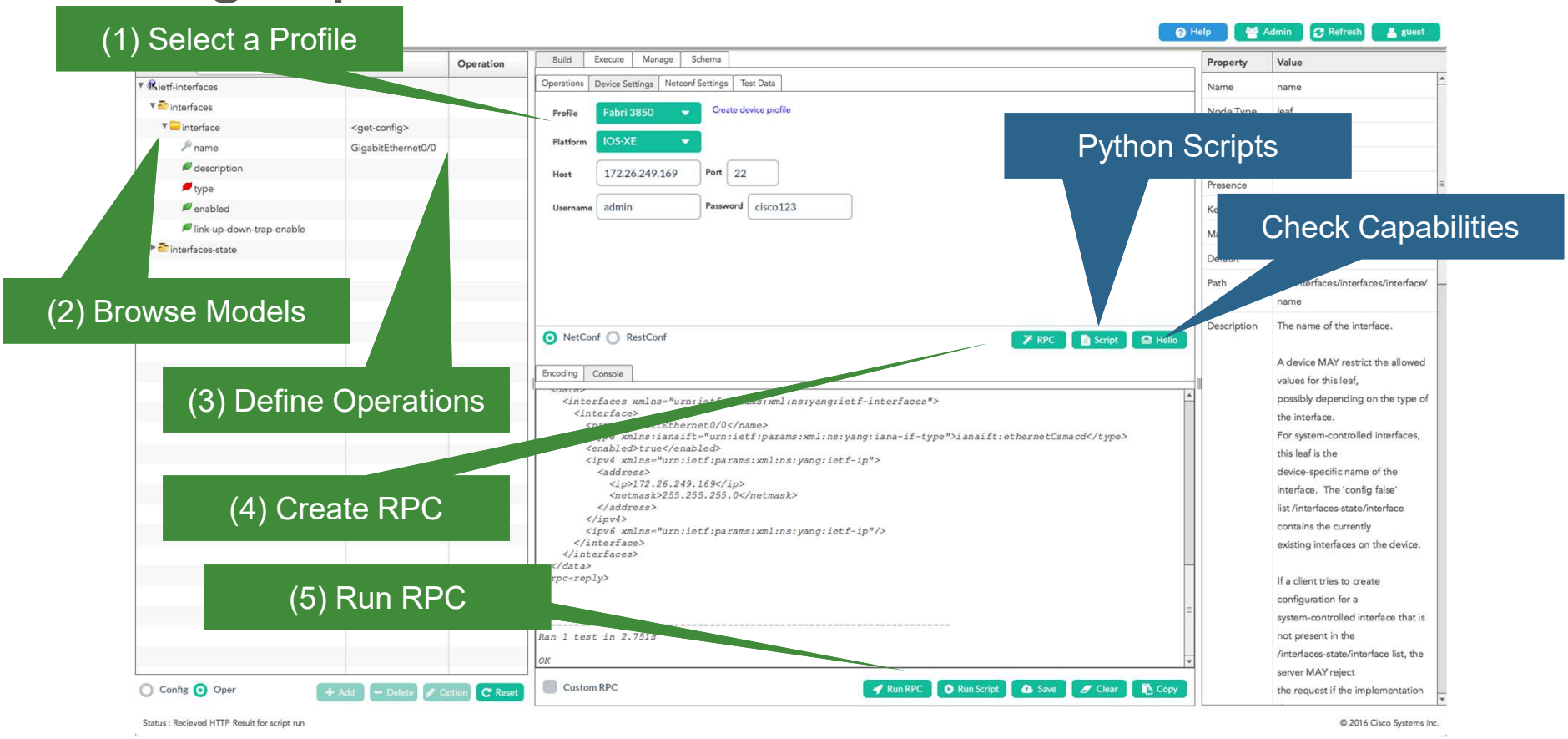
PYANG

$ **pyang -f tree ietf-interfaces.yang**

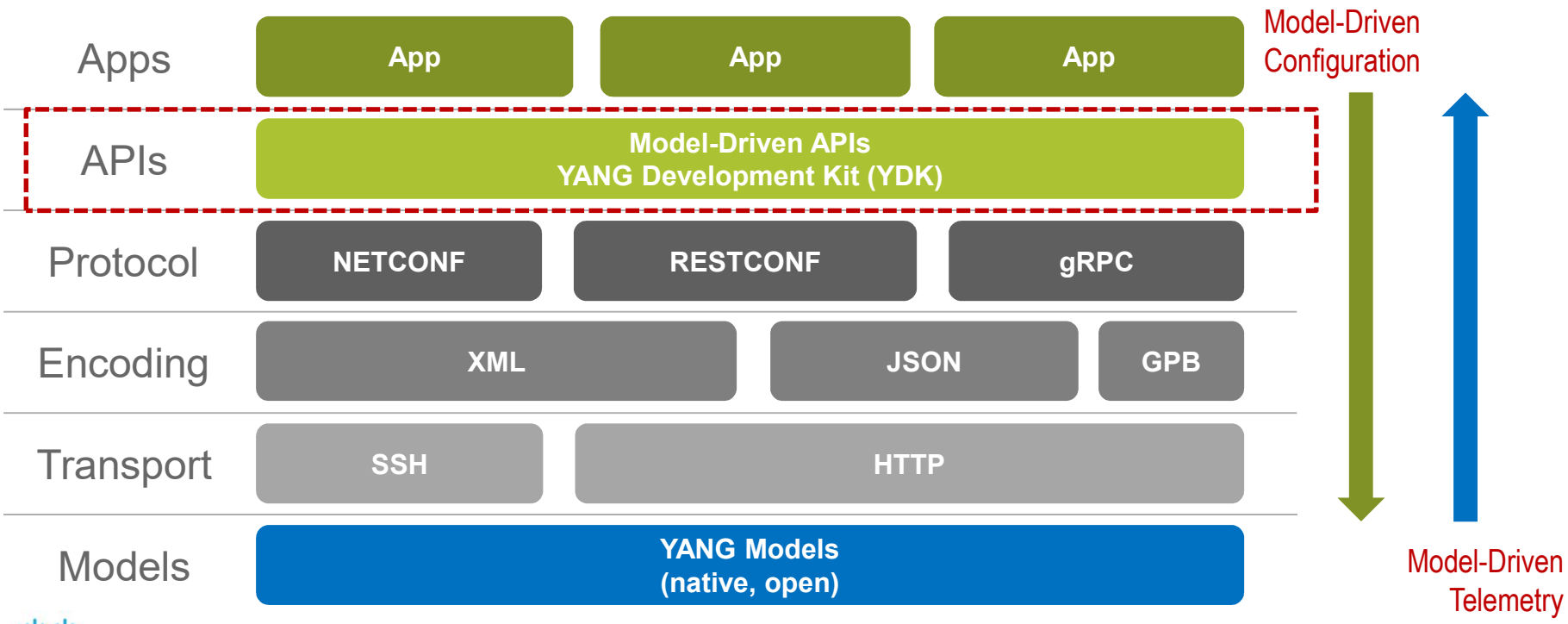
```
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                    string
   |     +--rw description?            string
   |     +--rw type                    identityref
   |     +--rw enabled?                boolean
   |     +--rw link-up-down-trap-enable?   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
         …
```

ひな形としてのデータモデルを
ツリー表示してくれる

YangExplorer

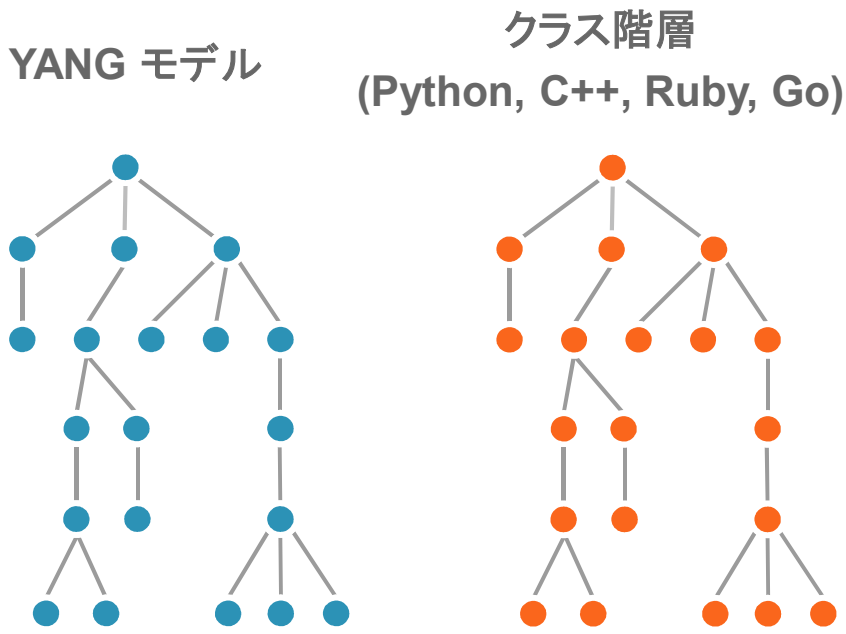


Model-Driven Programmability Stack



YDK(YANG Development Kit)
モデルドリブン API

- シンプルなアプリケーション開発
- YANGモデルとプログラミング言語のクラス階層を一対一でマッピング
- 抽象化されたコンフィグデータ
- YANG モデルからAPIを生成
- 強力な型バリデーション
- マルチ言語対応(Python, C++)



YDK-Py “Hello World” Using OC-BGP

```python
# Cisco YDK-Py OC-BGP “Hello world”
from ydk.services import CRUDService
from ydk.providers import NetconfServiceProvider
from ydk.models.openconfig import openconfig_bgp as oc_bgp

if __name__ == "__main__":
    provider = NetconfServiceProvider(address=10.0.0.1,
                                      port=830,
                                      username=“admin”,
                                      password=“admin”,
                                      protocol=“ssh”)
    crud = CRUDService()  # create CRUD service
    bgp = oc_bgp.Bgp()  # create oc-bgp object
    bgp.global_.config.as_ = 65000  # set local AS number
    crud.create(provider, bgp)  # create on NETCONF device
    provider.close()
    exit()
# End of script
```

```
module: openconfig-bgp
   +--rw bgp!
      +--rw global
      |  +--rw config
      |  |  +--rw as
      |  |  +--rw router-id?
      |  +--ro state
      |  |  +--ro as
      |  |  +--ro router-id?
      |  |  +--ro total-paths?
      |  |  +--ro total-prefixes?
...
```

Get your hands dirty with …

The Mission!

DevNet Express

DNA Programmability

cisco DevNet

developer.cisco.com

DevNet → Discover → Learning Tracks → DNA → Module LM-4602

https://learninglabs.cisco.com/tracks/devnet-express-dna-jp

ミッション:

NETCONF および Python を使用してプログラミングする

- pyang を使用してターゲット モデルを識別および調査する
- ncclient を利用してデバイスから運用データを取得する
- 返されたデータをユーザが使用しやすい形式に設定する
- 結果を Spark ルームにポストする

まとめ

学んだこと

- 標準デバイスインターフェイスの必要性
- YANG の理解
  - モデリング言語
  - モデル
  - データ
- NETCONF プロトコルがどのように動作するか
  - Pythonでどのように活用するか

What’s Next?

DevNet Express
DNA Programmability

cisco DevNet
developer.cisco.com