Model driven network programming made easy by open source

RIPE 78 Tutorial, 20 May 2019

Charles Eckel, Cisco DevNet
eckelcu@cisco.com, @eckelcu
Monday, 20 May 09:00 - 11:00

**Model driven network programming made easy by open source**
Charles Eckel

Software Defined Networking (SDN) started as the separation of the control plane and the data plane, but the true power of SDN lies in the ability to communicate with the network through well defined interfaces using standard protocols.

This tutorial provides a brief introduction to APIs and programmability in general, then dives into model driven network programmability and the role of YANG, NETCONF and RESTCONF. We then take a look at the wealth of open source and/or free software tools that exists to help master these technologies, including OpenDaylight, pyang, Postman, ncclient, YANG Development Kit (YDK), and YANG Explorer. We cover what they are, how to use them, and how to contribute back.

To get the most out of the tutorial and follow along with the hands-on exercises, you need a laptop with a development environment. You can follow these step-by-step instructions to setup your own development environment:

Note, access to online learning labs is free but requires a Cisco DevNet account, which can be setup easily using this RIPE 78 specific URL:
https://developer.cisco.com/join/ripe78

**Where:** Tutorial room
Agenda

• Setup
• Introduction to APIs
• REST APIs
• Network programmability
• Hands-on exercises
Introduction to Model Driven Programmability (ex: NETCONF/YANG)

Explore the reasons behind the move to Model Driven Programmability from traditional Interfaces such as CLI/SNMP.
Learn about the interaction between YANG data models and the new standard transport protocols of NETCONF and RESTCONF. Discover how to leverage NETCONF/RESTCONF to query and configure network devices.

1 Hour 30 Minutes

What and Why of Model Driven Programmability
What is “Model Driven Programmability” and why was it developed? What purpose do the new protocols and standards of YANG, NETCONF, and RESTCONF provide? Get the answers to these questions in this lab!

Introducing YANG Data Modeling for the Network
What’s YANG got to do with it? In this lab you’ll find out all about it! Learn about the YANG modeling language, checkout some of the available model options, and even see what network data looks like when fit into those models!

Exploring IOS XE YANG Data Models with NETCONF
Learn the ins and outs to working with NETCONF to access the YANG modeled configuration and operational data on your network devices. Get hands-on by initiating NETCONF connections, retrieving data, and sending configurations to the network.

Exploring IOS XE YANG Data Models with RESTCONF
So you want a REST API for the network? Well RESTCONF is your tool then. Checkout how YANG models become URLs with RESTCONF learn all there is to know about CRUD! You’ll explore RESTCONF with basic API calls and with Python!
What and Why of Model Driven Programmability

What is "Model Driven Programmability" and why was it developed? What purpose do the new protocols and standards of YANG, NETCONF, and RESTCONF provide? Get the answers to these questions in this lab!

Objectives

1. Understand the history that lead to model driven programmability
2. Understand how device features, data models, and transport protocols relate within a network element
3. Understand the purpose of a YANG Data Model, where they come from, and how to work with them.

Prerequisites

To complete this lab you need:

- A development environment with typical tools and applications. If you are at a DevNet Event using a provided workstation, you are ready to go. If you are working from your own workstation, please review the "How to setup your own computer" link at the top of this page.
- Lab infrastructure to target API calls and code. These labs and code examples are written to leverage the DevNet IOS XE Always On Sandbox. This lab is available for anyone to use, with only access to the Internet as a requirement. To use a different device, ensure the device is running IOS XE 16.6 or higher.
How To Setup Your Own Computer

Setting Up a Development Workstation for this Lab

Before beginning this lab on your workstation, you’ll want to install a standard set of development applications, tools, and interfaces. To learn more about what tools and what they offer, you can explore the What is a Development Environment, and why do you need one? Learning Lab.

To assist you with getting setup, DevNet has created Learning Labs that walk through the installation on different platforms.

- Setting up your Windows workstation as a development environment
- Setting up your MacOS workstation as a development environment
- Setting up your Linux (CentOS) workstation as a development environment

"Git" ting the Code and Setting Up the Local Environment

Now that your workstation is ready to go, the next step is to retrieve the code and install the lab specific requirements.

1. The code for this lab is available on GitHub at CiscoDevNet/dnav3-code.
2. Open a bash terminal and change to the directory where you would like to clone the repository. For example, a directory called code/ under your $HOME.

   ```bash
   cd ~/code
   ```

3. Clone the repository and change into the new folder.

   ```bash
   git clone https://github.com/CiscoDevNet/dnav3-code
cd dnav3-code
   ```
Introduction to APIs
Application Programming Interface

"It’s a way for two pieces of software to talk to each other"
For a time.. Humans were the only users
For a time.. Humans were the only users

Software displays results in User Interface (UI)

User asks for data or takes action by interacting with UI
But what about when the user is another software system….

Software returns results via API

Software asks for data or takes action by interacting with API
The API is the User Interface for software systems
APIs are sets of requirements that govern how one application can talk to another.
APIs help developers create apps that benefit the end user.

Google Maps returns map data via API.

Yelp asks for Map Data.

Users sees list of restaurants close to them.
“APIs are often referred to as “an engine of innovation.”

-- Programmable Web
REST APIs
REST Web service

• What is REST?
  – REp resentational S tate T ransfer (REST)
  – API framework built on HTTP

• What is a REST Web Service?
  – REST is an architecture style for designing networked applications.
  – Popular due to performance, scale, simplicity, and reliability
Request and Response, the REST API Flow

HTTP REQUEST
GET
https://devvie/api/hello
Request and Response, the REST API Flow
Request and Response, the REST API Flow

HTTP REQUEST
GET
https://devie/api/hello

HTTP RESPONSE
200 OK
JSON
< hello>
# HTTP Methods: What to do?

<table>
<thead>
<tr>
<th>HTTP Verb</th>
<th>Typical Purpose (CRUD)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>POST</strong></td>
<td>Create</td>
<td>Used to create a new object, or resource. Example: Add new book to library</td>
</tr>
<tr>
<td><strong>GET</strong></td>
<td>Read</td>
<td>Retrieve resource details from the system. Example: Get list of books from the library</td>
</tr>
<tr>
<td><strong>PUT</strong></td>
<td>Update</td>
<td>Typically used to replace or update a resource. Can be used to modify or create. Example: Update the borrower details for a book</td>
</tr>
<tr>
<td><strong>PATCH</strong></td>
<td>Update</td>
<td>Used to modify some details about a resource. Example: Change the author of a book</td>
</tr>
<tr>
<td><strong>DELETE</strong></td>
<td>Delete</td>
<td>Remove a resource from the system. Example: Delete a book from the library</td>
</tr>
</tbody>
</table>
# Response Status Codes: Did it work?

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Status Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>All looks good</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>New resource created</td>
</tr>
<tr>
<td>202</td>
<td>Accepted</td>
<td>Accepted for processing, but processing not completed</td>
</tr>
<tr>
<td>204</td>
<td>No Content</td>
<td>Request succeeded, but no message body returned</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request</td>
<td>Request was invalid</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>Authentication missing or incorrect</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>Request was understood, but not allowed</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>Resource not found</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Something wrong with the server</td>
</tr>
<tr>
<td>503</td>
<td>Service Unavailable</td>
<td>Server is unable to complete request</td>
</tr>
</tbody>
</table>
The URI: What are you Requesting?

https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1

- **Server or Host**
  - Resolves to the IP and port to which to connect

- **Resource**
  - The location of the data or object of interest

- **Parameters**
  - Details to scope, filter, or clarify a request. Often optional.

- **http:// or https://**
  - Protocol over which data is sent between client and server
  - ‘s’ in https stands for secure
Data: Sending and Receiving

- Contained in the message body
- GET responses will include a message body
- POST, PUT, PATCH requests typically include a message body
- Format typically JSON or XML
  - Check “Content-Type” header

```
{
  "success": true,
  "deck_id": "3p40paa87x90",
  "shuffled": true,
  "remaining": 52
}
```
## Headers:

### What additional details and metadata can I use?

<table>
<thead>
<tr>
<th>Header</th>
<th>Example Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type</td>
<td>application/json</td>
<td>Specify the format of the data in the body</td>
</tr>
<tr>
<td>Accept</td>
<td>application/json</td>
<td>Specify the requested format for returned data</td>
</tr>
<tr>
<td>Authorization</td>
<td>Basic dmFncmFudDp2YWdyYW50</td>
<td>Provide credentials to authorize a request</td>
</tr>
<tr>
<td>Date</td>
<td>Tue, 25 Jul 2017 19:26:00 GMT</td>
<td>Date and time of the message</td>
</tr>
</tbody>
</table>

- Used to pass information between client and server
- Included in both REQUEST and RESPONSE
- Some APIs use custom headers for authentication or other purpose
Review: Request/Response

GET /v1/people/me HTTPS/1.1
Host: api.ciscospark.com
Authorization: Bearer <redacted>
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36(KHTML, like Gecko) Chrome/49.0.2623.112 Safari/537.36

HTTP/1.1 200 OK
Date: Wed, 23 Jan 2019 23:12:11 GMT
Content-Type: application/json;charset=UTF-8
Content-Encoding: gzip
Content-Length: 323
Trackingid: ROUTER_5C48F4B1-9789-01BB-4148-xxxxxxxxx
Vary: Accept-Encoding
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload

{
"id": "Y2lzY29zcGFyazovL3VzL1BFT1BMRS9iODBjM2NmOC01ZGIwLTQyNzAtOThiZS1mYzFhYjA3MzE1YWE",
"emails": ["eckelcu@cisco.com"],
"displayName": "Charles Eckel",
"nickName": "Charles",
"firstName": "Charles",
"lastName": "Eckel",
"status": "active",
"type": "person"
}
Many Options for Working with REST APIs

• Web browser
  • Chrome, Firefox, etc.
• curl
  • Linux command line application
• Postman
  • API testing application and framework
• Requests
  • Python library for scripting
• OpenAPI/Swagger
  • Dynamic API Documentation
Web Browser

https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1

<table>
<thead>
<tr>
<th>JSON</th>
<th>Raw Data</th>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Copy</td>
<td>Collapse All</td>
</tr>
<tr>
<td>success:</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>shuffled:</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>deck_id:</td>
<td>&quot;86p8aq57r7y7&quot;</td>
<td></td>
</tr>
<tr>
<td>remaining:</td>
<td>52</td>
<td></td>
</tr>
</tbody>
</table>
Web Browser

https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1

```json
{"success": true, "shuffled": true, "deck_id": "86p8aq57r7y7", "remaining": 52}
```
Web Browser

https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1

<table>
<thead>
<tr>
<th>Request Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
</tr>
<tr>
<td>Accept-Encoding</td>
</tr>
<tr>
<td>Accept-Language</td>
</tr>
<tr>
<td>Connection</td>
</tr>
<tr>
<td>DNT</td>
</tr>
<tr>
<td>Host</td>
</tr>
<tr>
<td>Upgrade-Insecure-Requests</td>
</tr>
<tr>
<td>User-Agent</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Firefox-Spdy</td>
</tr>
<tr>
<td>access-control-allow-origin</td>
</tr>
<tr>
<td>cf-ray</td>
</tr>
<tr>
<td>content-encoding</td>
</tr>
<tr>
<td>content-type</td>
</tr>
<tr>
<td>date</td>
</tr>
<tr>
<td>server</td>
</tr>
<tr>
<td>x-frame-options</td>
</tr>
</tbody>
</table>
curl

$ curl
https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1

{"success": true, "shuffled": true, "deck_id": "sr405eihisjl", "remaining": 52}
Postman

GET Deck of cards - new deck

Deck of cards - new deck

GET https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1...

**Params**

- **KEY**: deck_count
- **Value**: 1...

**Body**

```
{
  "success": true,
  "shuffled": true,
  "deck_id": "ouzlwi8j195h",
  "remaining": 52
}
```
Postman
```python
import requests
url = "https://deckofcardsapi.com/api/deck/new/shuffle/
querystring = {"deck_count":"1000"}
payload = ""
headers = {
    'cache-control': "no-cache",
    'Postman-Token': "4ed7aef4-90e0-46b0-bc3a-45c575a7c000"
}
response = requests.request("GET", url, data=payload, headers=headers, params=querystring)
print(response.text)
```
Network programmability
Why Network Programmability Matters

**Network Expenses**

- **CAPEX**: 33%
- **OPEX**: 67%

Source: Forrester

**Deployment Speed**

- **Computing**
- **Networking**

Seconds: 0, 10, 100, 1000

Source: Open Compute Project
The Need for Something Better

- SNMP had failed
  - For configuration, that is
  - Extensive use in fault handling and monitoring
- CLI scripting
  - “Market share” 70%+

RFC 3535

Abstract

This document provides an overview of a workshop held by the Internet Architecture Board (IAB) on Network Management. The workshop was hosted by CNRI in Reston, VA, USA from June 4 thru June 6, 2002. The goal of the workshop was to continue the important dialog started between network operators and protocol developers, and to guide the IETFs focus on future work regarding network management.
Best Practices Coming Together

- CLI Best Practices
- SNMP Experience
- Operations Requirements
- NETCONF, RESTCONF and YANG
YANG
**YANG**

Data Modeling Language for Networking

- Modeling language, YANG version 1 [RFC6020], YANG version 1.1 [RFC7950]
- Models configuration and state data, RPCs, and notifications
- Defines semantics
  - Constraints (i.e. “MUSTs”)
  - Reusable structures
  - Built-in and derived types

YANG is a full, formal contract language with rich syntax and semantics for network data
YANG Model Example

• Screenshot from ietf-interfaces.yang

• Container ‘interfaces' with list of interface' items

• List items (leafs) have a ‘name' which is also the key for the list

```
container interfaces {
  description
  "Interface configuration parameters.";

  list interface {
    key "name";

    description
    "The list of configured interfaces on the device.

    The operational state of an interface is available in the
    /interfaces-state/interface list. If the configuration of a
    system-controlled interface cannot be used by the system
    (e.g., the interface hardware present does not match the
    interface type), then the configuration is not applied to
    the system-controlled interface shown in the
    /interfaces-state/interface list. If the configuration
    of a user-controlled interface cannot be used by the system,
    the configured interface is not instantiated in the
    /interfaces-state/interface list.";

    leaf name {
      type string;
      description
      "The name of the interface.

      A device MAY restrict the allowed values for this leaf,
      possibly depending on the type of the interface.
      For system-controlled interfaces, this leaf is the
      device-specific name of the interface. The 'config false'
      list /interfaces-state/interface contains the currently
      existing interfaces on the device.
    }
  }
}
```
Finding YANG Models
https://github.com/YangModels/
Tools to work with YANG Models

- **pyang** - An extensible YANG validator and converter
  - Command line tool
  - Source Code - [https://github.com/mbj4668/pyang](https://github.com/mbj4668/pyang)
  - Python Package - [https://pypi.python.org/pypi/pyang](https://pypi.python.org/pypi/pyang)

- **YANG Catalog** - YANG validator, search, and impact tools
  - Web Based
  - [https://yangcatalog.org/](https://yangcatalog.org/)

- **OpenDaylight YANG Tools**
  - Tools supporting NETCONF and YANG
  - Code generation from YANG models
  - [https://wiki.opendaylight.org/view/YANG_Tools:Main](https://wiki.opendaylight.org/view/YANG_Tools:Main)
$ pyang -f tree <yang-file>
Yang Catalog

https://yangcatalog.org/yang-search/
Building a Plugin/Application with OpenDaylight YANG tools

1. Generate APIs
   - Yang Tools
     - Yang Model
     - Generated API Definition

2. Create API Bundle
   - Maven Build Tools
     - "API" OSGI Bundle

3. Create Plugin Bundle
   - Maven Build Tools
     - "Plugin" OSGI Bundle

4. Deploy
   - "API" OSGI Bundle
   - "Plugin" OSGI Bundle

Controller
NETCONF
NETCONF

IETF network management protocol

- Connection oriented, with transport via SSH/TSL
- Data defined by YANG models, encoded in XML
- Distinguishes between configuration and state data
- Multiple configuration datastores (candidate, running, startup)
- Change validation, transactions, filtering, and notifications

NETCONF provides fundamental programming features for convenient and robust automation of network services
NETCONF Sessions

- NETCONF is connection-oriented
  - SSH, TLS as underlying transport
  - XML for payload
- NETCONF client establishes session with server
- Session establishment: <hello> exchange
  - Announce capabilities, modules, features
- Session termination
  - <close-session>, <kill-session>
NETCONF Commands

- get : to retrieve operational data
- get-config : to retrieve configuration data
- edit-config : to edit a device configuration
- copy-config : to copy a configuration to another data store (e.g. non-volatile memory)
- delete-config : to delete a configuration in a data store
DevNet Always On Sandbox

- CSR1000V Host: `ios-xe-mgmt.cisco.com`
- SSH Port: 8181
- NETCONF Port: 10000
- RESTCONF Port: 9443 (HTTPS)

- Credentials:
  - Username: `root`
  - Password: `D_Vay!_10&`
Connect to DevNet Always on Sandbox

ssh root@ios-xe-mgmt.cisco.com -p 8181
ssh -oHostKeyAlgorithms=+ssh-dss root@ios-xe-mgmt.cisco.com -p 10000 -s netconf

Welcome to the DevNet Always On Sandbox for IOS XE

This is a shared sandbox available for anyone to use to test APIs, explore features, and test scripts. Please keep this in mind as you use it, and respect others use.

The following programmability features are already enabled:
- NETCONF
- RESTCONF

Thanks for stopping by.

csr1000v#show run
Building configuration...

Current configuration : 5332 bytes
!
! Last configuration change at 16:55:51 UTC Fri May 17 2019 by root
!
version 16.8
service timestamps debug datetime msec
service timestamps log datetime msec
platform qfp utilization monitor load 80
no platform punt-keepalive disable-kernel-core
platform console virtual
!
hostname csr1000v
NETCONF using ncclient – Python code

https://developer.cisco.com/learning/modules/intro-device-level-interfaces/intro-netconf/step/1
NETCONF using ncclient - Output

(venv) ECKELCU-M-H15L:netconf eckelcu$ python get_interface_list.py
Opening NETCONF Connection to ios-xe-mgmt.cisco.com

Sending a <get-config> operation to the device.

Here is the raw XML data returned from the device.

```xml
<?xml version="1.0" ?>
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>GigabitEthernet1</name>
        <description>DON'T TOUCH ME</description>
      </interface>
      ...snip...
      <interface>
        <name>Tunnel2</name>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

The interface status of the device is:
- Interface GigabitEthernet1 enabled status is true
- Interface GigabitEthernet2 enabled status is true
- Interface GigabitEthernet3 enabled status is false
- Interface Loopback0 enabled status is true
- Interface Tunnel0 enabled status is true
- Interface Tunnel1 enabled status is true
- Interface Tunnel2 enabled status is true
RESTCONF

Restful API for YANG data models

- IETF RFC 8040
- Configuration and state data exposed as resources
- Access data using REST verbs (GET / PUT / POST …)
- Construct URIs, based on structure of YANG model, to access data
- HTTP instead of SSH for transport
- JSON in addition to XML for data encoding

RESTCONF provides light weight interface to network datastores leveraging well known combination of REST and JSON
GET
https://{host}:{port}/restconf/data/ietf-interfaces:interfaces-state/interface=GigabitEthernet1

"ietf-interfaces:interface": {
    "name": "GigabitEthernet1",
    "type": "iana-if-type:ethernetCsmacd",
    "admin-status": "up",
    "oper-status": "up",
    "last-change": "2019-05-16T19:40:02.000393+00:00",
    "if-index": 1,
    "phys-address": "00:50:56:bb:18:c4",
    "speed": "102400000",
    "statistics": {
        "discontinuity-time": "2019-05-16T19:38:03.000573+00:00",
        "in-octets": "5339802",
        "in-unicast-pkts": "48925",
        "in-broadcast-pkts": "0",
        "in-multicast-pkts": "0",
        "in-discards": 0,
        "in-errors": 0,
        "in-unknown-protos": 0,
        "out-octets": "9405098",
        "out-unicast-pkts": "17451",
        "out-broadcast-pkts": "0",
        "out-multicast-pkts": "0",
        "out-discards": 0,
        "out-errors": 0
    }
}
High Level Manageability Architecture

Application
- ANY (C, Java, Python)
- NETCONF client
- RESTCONF client
- ANY (Java, Python, Perl, PHP)

Transport
- YANG-based XML
- SSH / TLS
- YANG-based XML/JSON
- HTTPS

Network Device
- NETCONF server
- RESTCONF server
- Manageability Infra
- Config DB
- BGP
- QoS
- VXLAN

Manageability
- Infra
- Config
- DB

YANG-based XML/JSON
- ANY (Java, Python, Perl, PHP)

SSH / TLS
- ANY (C, Java, Python)

HTTPS
- ANY (Java, Python, Perl, PHP)
RESTCONF with curl

The Request

$ curl -vk \
  -u root:D_Vay\! 10\& \
  -H 'accept: application/yang-data+json' \

> GET /restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1 HTTP/1.1
> Host: ios-xe-mgmt.cisco.com:9443
> Authorization: Basic cm9vdDpEX1ZheSFfMTAm
> User-Agent: curl/7.54.0
> accept: application/yang-data+json
>

- **-u** provides **user:password** for Basic Authentication
- **-H** to set headers
- Lines beginning with “>” indicate Request elements
- Lines beginning with “<” indicate Response elements (next slide)
RESTCONF with curl

The Response - Headers

- HTTP/1.1 200 OK
- Server: nginx
- Date: Fri, 25 Jan 2019 17:37:43 GMT
- Content-Type: application/yang-data+json
- Transfer-Encoding: chunked
- Connection: close
- Cache-Control: private, no-cache, ...
- Pragma: no-cache

The Response - Body

```json
{
   "ietf.interfaces:interface": {
      "name": "GigabitEthernet1",
      "description": "DON'T TOUCH ME",
      "type": "iana-if-type:ethernetCsmacd",
      "enabled": true,
      "ietf-ip:ipv4": {
         "address": [
            {
               "ip": "10.10.20.48",
               "netmask": "255.255.255.0"
            }
         ],
         "ietf-ip:ipv6": {
            
         }
      }
   }
}
```
RESTCONF with Postman
OpenDaylight YANG UI
Questions?
Thank you!