AIS Hackathon
Network Programmability
May 2018

Charles Eckel, Cisco DevNet
eckelcu@cisco.com
Agenda

• What is SDN
• What is OpenDaylight
• Network programmability
• Hands-on Exercises with Cisco IOS XE
• Installing OpenDaylight
• Example use cases
• Hands-on Exercises with OpenDaylight
• Conclusions
What is SDN
Software Defined Networking (SDN)

- Control & Data Planes separation?
  - OpenFlow?
  - Logically centralized control Plane?
  - White label switches?

- This a valid & useful SDN use case, but...

- SDN can be defined more broadly:
  - Network is a source of vast amount of data...
  - ..that can be utilized by variety of SDN applications

- True power of SDN is network programmability
SDN - A Broader Definition

- Application Developer Environment
- Management and Orchestration
- Analysis and Monitoring, Performance and Security
- Network Services
- Control Plane
- Forwarding Plane
- Network Elements and Abstraction
- Transport

Generic feedback/control/policy loop between apps and the network

Harvest Network Intelligence

Program for Optimized Experience
What Do We Need from an SDN Controller?

• A platform for deploying SDN applications
• Provide an SDN application development environment
  • Developer-friendly APIs to network elements (REST/JSON, pub/sub, etc.)
  • Network-level abstraction through topologies
  • Protocol independence for network-facing applications
What is OpenDaylight
The OpenDaylight Community

- Founded in February 2013
- Run by the Linux Foundation
- Eclipse Public License
- 15 founding companies provided software and developers
- 600+ contributors
- 2.5M+ lines of code
- Mostly Java

- First release “Hydrogen”
  - February 2014
- Release frequency
  - Roughly every 6 months
- Current release - “Nitrogen”
  - 7th release, Sept 26, 2017
  - SR1 released Nov 26, 2017
- Next release is Oxygen
  - March 2018
Software Architecture

- Java - enterprise-grade, cross-platform compatible language
- Java Interfaces - for event listening, specifications and forming patterns
- Maven – build system
- Karaf – based on OSGi, provides:
  - dynamic loading of bundles
  - registering dependencies and services exported
  - exchanging information across bundles

![OSGi Framework (Equinox) Diagram]

MD-SAL  Feature A  Feature B  Feature X

Karaf

OSGi Framework (Equinox)
Network programmability
Why Network Programmability Matters

Network Expenses

- CAPEX: 33%
- OPEX: 67%

Source: Forrester

Deployment Speed

- Computing: 0 seconds
- Networking: 1000 seconds

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 IETF's focus on future work regarding network management.
Best Practices Coming Together

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

Data Modeling Language for Networking

- Modeling language, defined in RFC 6020
- Models configuration and state data, RPCs, and notifications
- Defines semantics
  - Constraints (i.e. “MUSTs”)
  - Reusable structures
  - Built-in and derived types

In Summary:
YANG is a full, formal contract language with rich syntax and semantics for network data
Model Structure

- Data structured as a tree
- Main node types:
  - Container
  - List
  - Leaf List
  - Leaf
YANG Model Example

- Screenshot from `network-topology.yang`
- Container 'network-topology' with list of 'topology' items
- List items (leafs) have a 'topology-id' which is also the key for the list
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 Explorer** - YANG Browser and RPC Builder
  - Web Based GUI
  - [https://github.com/CiscoDevNet/yang-explorer](https://github.com/CiscoDevNet/yang-explorer)

- **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)
Display a YANG Module

$ pyang -f tree <yang-file>
pyang Tip – JavaScript Tree Output

• Use `pyang -f jstree -p <model.yang> -o <output.html>`

• Produces collapsible Tree / HTML
Building a Plugin/Application with YANG tools
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

In Summary:
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
RESTCONF
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

In Summary:
RESTCONF provides light weight interface to network datastores leveraging well known combination of REST and JSON
RESTCONF URI & JSON Example

```xml
<node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
  <node-id>vpp1</node-id>
  <host xmlns="urn:opendaylight:netconf-node-topology">{{vpp1_address}}</host>
  <port xmlns="urn:opendaylight:netconf-node-topology">2831</port>
  <username xmlns="urn:opendaylight:netconf-node-topology">admin</username>
  <password xmlns="urn:opendaylight:netconf-node-topology">admin</password>
  <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
  <keepalive-delay xmlns="urn:opendaylight:netconf-node-topology">0</keepalive-delay>
</node>
```
High Level Manageability Architecture

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

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

Config DB

Manageability Infra

BGP

QoS

VXLAN
Mounting YANG Datastores

OpenDaylight NETCONF Node “Discovery”

- Nodes added by POSTing to config:modules
- OpenDaylight connects to each node
- OpenDaylight learns capabilities (YANG modules) and stores to model cache
  - Cache at ~/cache/schema. Filenames of form yang-model@2016-07-12.yang.
Hands-on Exercises
https://learninglabs.cisco.com/modules/networking-basics
Intro to Coding Fundamentals
Get started with coding basics by learning the fundamentals of coding with Python and parsing JSON.

© 2 Hours 15 Minutes

- **Introduction to Git**
  Learn the basics of git and how to clone an online repository to a local machine.

- **Python Primer Level 1**
  Learn the basics of Python syntax, operators, conditional statements, and functions.

- **Python Primer Level 2**
  Learn the how to use libraries, virtual environments, loops, nested datatypes and about execution flow.

- **Using Python to Parse JSON**
  Learn the basics of using Python to parse JSON.

https://learninglabs.cisco.com/modules/fundamentals
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 URIs with RESTCONF learn all there is to know about CRUD! You'll explore RESTCONF with basic API calls and with Python!
Accessing DevNet Sandbox to Reserve Your Own Setup

https://devnetsandbox.cisco.com/RM/Topology
Reserve Same Setup as Used in Learning Lab
IOS XE Programmability
Installing OpenDaylight
## Distributions

[https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation](https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation)

### Downloads

<table>
<thead>
<tr>
<th>Release</th>
<th>Release date</th>
<th>Downloads</th>
<th>Documentation</th>
</tr>
</thead>
</table>
| Carbon SR2      | October 16, 2017 | • Pre-Built Tar  
• Pre-Built Zip  
• NeXT UI  
• Virtual Tenant Network (VTN) Coordinator | • Getting Started Guide  
• Developers Guide  
• User Guide  
• Installation Guide  
• Using OpenDaylight with OpenStack  
• Release Notes |
| Nitrogen SR1     | November 26, 2017| • Pre-Built Tar  
• Pre-Built Zip  
• Virtual Tenant Network (VTN) Coordinator  
• OpFlex | • Getting Started Guide  
• Developers Guide  
• User Guide  
• Installation Guide  
• Using OpenDaylight with OpenStack  
• Release Notes |
$ unzip karaf-0.7.1.zip
Archive: karaf-0.7.1.zip
    creating: karaf-0.7.1/system/ ...
$ cd karaf-0.7.1
$ ./bin/karaf
karaf: Enabling Java debug options: -Xdebug -Xnoagent -Djava.compiler=NONE
    -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005
Listening for transport dt_socket at address: 5005
Apache Karaf starting up. Press Enter to open the shell now...
100% [========================================================================]
Karaf started in 0s. Bundle stats: 10 active, 10 total

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
Install Features using Karaf

• OpenDaylight distro comes without any features enabled by default

• All features are available for you to install
  • `feature:list` list all features available
  • `feature:list -i` list all features installed
  • `feature:install <feature>` install the `<feature>` feature
  • `feature:install <feature-1> <feature-2> ... <feature-n>` install list of features
  • `feature:uninstall <feature>` uninstalls the `<feature>` feature
Install DLUX, NETCONF, and RESTCONF

```bash
opendaylight_user@root> feature:install odl-dlux-core
opendaylight_user@root> feature:install odl-dluxapps-yangui
opendaylight_user@root> feature:install odl-restconf-all
opendaylight_user@root> feature:install odl-netconf-all
opendaylight_user@root> feature:install odl-netconf-topology
Opendaylight_user@root> feature:install odl-netconf-connector-ssh
opendaylight_user@root> feature:list -r
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Required</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl-netconf-topology</td>
<td>1.3.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-restconf-all</td>
<td>1.6.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-netconf-connector-ssh</td>
<td>1.3.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-dluxapps-yangui</td>
<td>0.6.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-netconf-all</td>
<td>1.3.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-dlux-core</td>
<td>0.6.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>wrap</td>
<td>0.0.0</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>standard</td>
<td>4.0.10</td>
<td>x</td>
<td>Started</td>
</tr>
</tbody>
</table>
http://localhost:8181/index.html#/yangui/index
Example Use Cases
Mininet, OVSDB and OpenFlow
Cisco IOS XR using BGP-LS and PCE-P

- Cisco XRv topology in dCloud
  - dCloud is http://dcloud.cisco.com (requires CCO login)
  - “OpenDaylight Boron SR3 with Apps with 8 Nodes v1”
  - ODL runs in dCloud (or use anyconnect/openconnect VPN to use local ODL instance)

- Use Pathman-SR application to create Segment Routed LSPs
VPP/Honeycomb using NETCONF and RESTCONF

• VPP is a high-performance, open source, software forwarder
  • [http://www.fd.io](http://www.fd.io)

• Honeycomb provides NETCONF and RESTCONF interfaces to VPP
OpenDaylight with Mininet – Step by Step

• Install, setup, and start Mininet VM using VirtualBox
  • Great instructions at http://www.brianlinkletter.com/set-up-mininet/
  • Login (user=mininet, password=mininet)

• Within OpenDaylight, enable required feature set
  • opendaylight-user@root> feature:install odl-l2switch-switch
    odl-dlux-core odl-dluxapps-applications

• Within Mininet VM, start 3 switches controlled by OpenDaylight
  • mininet@mininet-vm:~$ sudo mn --topo linear,3 --mac
    --controller=remote,ip=<OpenDaylight-IP>,port=6633
    --switch ovs,protocols=OpenFlow13
  • mininet@mininet-vm:~$ pingall

• From browser, log into OpenDaylight DLUX
  • http://<OpenDaylight-IP>:8181/index.html
    (credentials: admin/admin)
Mininet Network Start

```
[mininet@mininet-vm:~]$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.40.18,
port=6633 --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
[mininet]> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> 
```
Using DLUX

• From Browser, log into OpenDaylight DLUX
  • http://<OpenDaylight-IP>:8181/index.html (credentials: admin/admin)

• Check out the network and switches by clicking on Nodes, Node Connectors
REST APIs

- Click on Yang UI and Expand All to see the REST APIs available
Inventory of Network Nodes

- GET opendaylight-inventory -> operational -> nodes
VPP/Honeycomb using NETCONF and RESTCONF

Step by Step

1. Create VM for Honeycomb and VPP
2. Install VPP and Honeycomb on VM
3. Start VPP and Honeycomb
4. Connect to VPP using CLI
5. Add interface(s) to VPP
6. Connect to VPP using Honeycomb/NETCONF
7. Connect to VPP using OpenDaylight
VPP/Honeycomb using NETCONF and RESTCONF

1. Create VM for Honeycomb and VPP

   • Download minimal CentOS 7 from https://www.centos.org/download/

   • Create VM and enable ssh using http://www.jeramysingleton.com/install-centos-7-minimal-in-virtualbox/ to create VM and enable ssh
     • Add two host-only adapters with DHCP and promiscuous mode enabled
       • One for VPP, another to access Honeycomb directly from laptop
     • To add sudo for my user (devnet/devnet) using https://www.digitalocean.com/community/tutorials/how-to-create-a-sudo-user-on-centos-quickstart
VPP/Honeycomb using NETCONF and RESTCONF

2. Install VPP and Honeycomb on VM

- FD.io wiki provides instructions for installing VPP and installing HC
  - Add the FD.io repo:
    - Add the following lines to /etc/yum.repos.d/honeycomb-release.repo
      [honeycomb-release]
      name=honeycomb release branch latest merge
      baseurl=https://nexus.fd.io/content/repositories/fd.io.centos7/
      enabled=1
      gpgcheck=0
  - Install both packages
    - sudo yum install vpp
    - sudo yum install honeycomb
VPP/Honeycomb using NETCONF and RESTCONF

3. Start VPP and Honeycomb

- Reset iptables
  - `sudo ./iptables-reset.sh`

- Flush interface to be used for DPDK
  - `sudo ifconfig enp0s8 down`
  - `sudo ip add flush dev enp0s8`

- Start VPP, then Honeycomb
  - `sudo service vpp start`
  - `sudo service honeycomb start`

- Check availability of Honeycomb’s SSH/NETCONF port:
  - `netstat -an | grep 2831`
4. Connect to VPP Using CLI

- $ ssh devnet@192.168.60.101
- $ sudo vppctl

```
/___/
/ / /
/ /  
/ /   
/ /    
/ /     
/ /      
/ /     
/ /      
/ /     
/ /    
/ /  
/ / /
/ /  
/ /   
/ /    
/ /     
/ /      
/ /     
/ /    
/ /  
/ / /
/ /  
```

- $vpp# show interface

<table>
<thead>
<tr>
<th>Name</th>
<th>Idx</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigabitEthernet0/8/0</td>
<td>1</td>
<td>down</td>
</tr>
<tr>
<td>local0</td>
<td>0</td>
<td>down</td>
</tr>
</tbody>
</table>
VPP/Honeycomb using NETCONF and RESTCONF

5. Add interface(s) to VPP

- Add a virtual interface using [https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise:_Create_an_Interface](https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise:_Create_an_Interface)

- Optionally add a physical interface using [https://wiki.fd.io/view/VPP/How_To_Connect_A_PCI_Interface_To_VPP](https://wiki.fd.io/view/VPP/How_To_Connect_A_PCI_Interface_To_VPP)
  - Need to have associated a host-only network; if none, add one with DHCP and promiscuous mode before proceeding, should get something like
  - Details in notes section of slide
VPP/Honeycomb using NETCONF and RESTCONF

6. Connect to VPP Using Honeycomb and NETCONF

- Honeycomb listens on port 2831 for SSH/NETCONF
- Connect to VPP and issue for sample commands using: https://wiki.fd.io/view/Honeycomb/Releases/1609/Running_Honeycomb
- You also need to add ssh-dss when connecting via ssh
  - $ ssh -oHostKeyAlgorithms=+ssh-dss admin@192.168.60.101 -p 2831 -s netconf
- By default, honeycomb listens for RESTCONF on localhost:2831. To connect via RESTCONF from off-box
  - $ sudo vi /opt/honeycomb/config/restconf.json
    - Change restconf config from localhost or 127.0.0.1 to 0.0.0.0, e.g.
      "restconf-binding-address": "0.0.0.0",
      "restconf-port": 8183,
VPP/Honeycomb using NETCONF and RESTCONF

7. Connect to VPP Using OpenDaylight

• Import Postman environment

• Import Postman collection

• Add VPP to OpenDaylight topology with Postman
  • PUT
    http://{{odl_address}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1

• View configuration in OpenDaylight DLUX
### Collections

- **ODL PCEP**: 9 requests
- **ODL XR Netconf**: 52 requests
- **ODL-VPP**: 7 requests

#### Add VPP1

- **GET**: Get NETCONF Topology
- **GET**: List ifcs - cf
- **GET**: List ifcs - oper
- **GET**: List ifcs host-gigabit-ethernet
- **PUT**: Enable local0 interface - cf
- **PUT**: Enable gigabit-ethernet interface - cf

#### Add VPP1

**PUT**

```
http://{{odl_address}}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1
```

**Authorization**

- **Basic WRTaw66YWRtaW4=**

**Accept**

```
application/xml
```

**Content-Type**

```
application/xml
```

**Body**

- **Cookies (1)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Domain</th>
<th>Path</th>
<th>Expires</th>
<th>HTTP</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSESSIONID</td>
<td>1ap8828t7p7w1rge02pwm16</td>
<td>localhost</td>
<td>/restconf</td>
<td></td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
### Get NETCONF Topology

**GET**


**Authorization**

- Content-Type: application/xml
- Authorization: Basic YWRtaW46YWUtcg==

**Body**

```
{
    "topology": [
        {
            "topology-id": "topology-netconf",
            "node": [
                {
                    "node-id": "controller-config",
                    "netconf-node-topology:available-capabilities": {
                        "available-capability": [
                            {
                                "capability": "urn:ietf:params:netconf:capability:candidate:1.0",
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

**Response**

- **Status:** 200 OK
- **Time:** 47 ms
- **Size:** 26.92 KB
Request sent successfully

- node-list
- node <node-id: vpp1>
  - node-id: vpp1
  - host: 192.168.60.101
  - port: 2831
  - connection-status: connected
Hands on Exercises
Nitrogen SR1 Sandbox in Cisco dCloud: https://dcloud.cisco.com/

Catalog

Sort By: Published Date

1 result in: "opendaylight nitrogen sr1"

OpenDaylight Nitrogen SR1 with Apps with 8 Nodes v1

ID: 399579  Published Date: 18-Jan-2018 05:27  Demonstration  Service Provider

Demonstrate how to accelerate process adoption, foster innovation, reduce risk, and create a more transparent approach to SDN by using OpenDaylight.

Schedule
Overview

OpenDaylight (ODL) is a collaborative, open-source project used to advance software-defined networking (SDN). OpenDaylight is a community-led, industry-supported framework consisting of code and blueprints. Using this framework, you can accelerate process adoption, foster innovation, reduce risk, and create a more transparent approach to SDN. OpenDaylight can be a core component within any SDN architecture. Building on open-source SDN and NFV controllers enables users to reduce operational complexity, extend the life of their existing infrastructure hardware, and enable new services and capabilities only available with SDN.

Scenarios

- Scenario 1: Explore ODL Features
- Scenario 2: Explore DLUX
- Scenario 3: Install BGP Pathman Application
- Scenario 4: Enable OpenFlow in Karaf
- Scenario 5: Install OpenFlow Manager Application
- Scenario 6: Explore Pathman Segment Routing
- Scenario 7: Explore netACL Application
- Scenario 8: Explore Yangman
Conclusions
Key Takeaways

• SDN is more than just OpenFlow
• Network programmability is key benefit of SDN
• OpenDaylight provides a platform for network applications and programmable network infrastructure
Thank you!