OpenDaylight as a Platform for Network Programmability

Charles Eckel, Open Source Developer Evangelist, Cisco DevNet
@eckelcu, eckelcu@cisco.com
Agenda

• What is SDN?
• What is OpenDaylight?
• Network Programmability
• Installation
• Example Use Cases
• Additional Resources
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

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

Harvest Network Intelligence

Application Developer Environment

Management and Orchestration

Analysis and Monitoring, Performance and Security

Network Services

Control Plane

Forwarding Plane

Network Elements and Abstraction

Transport

Program for Optimized Experience

OpenDaylight as a Platform for Network Programmability
What Do We Need from an SDN Controller?

• A platform for deploying SDN applications

• Provide an SDN application development environment
  1. Developer-friendly APIs to network elements (REST/JSON, pub/sub, etc.)
  2. Network-level abstraction through topologies
  3. 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
  • Every 4-6 months
• Current release - “Carbon”
  • Sixth release, May 26, 2017
  • SR1 released July 14, 2017
• Next release is Nitrogen
  • September 2017
Software Architecture

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

OSGi Framework (Equinox)

Feature A
Feature B
Feature X
MD-SAL
Karaf
Network Programmability
Why Network Programmability Matters

**Network Expenses**

- CAPEX: 33%
- OPEX: 67%

Source: Forrester

**Deployment Speed**

- Computing: 0 seconds
- Networking: 1000 seconds

Source: Open Compute Project

OpenDaylight as a Platform for Network Programmability  © 2017 Cisco and/or its affiliates. All rights reserved.  Cisco Public
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

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

Data Modeling Language for Networking

• Modeling language, defined in RFC 6020
• Represents operational state, configuration, transactions, 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

OpenDaylight as a Platform for Network Programmability
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

```yang
/*
 * Configuration data nodes
 */

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."
```
Tools to work with YANG Models

• **pyang** - An extensible YANG validator and converter in python
  • 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)
  • Command line tool

• **YANG Explorer** - YANG Browser and RPC Builder
  • [https://github.com/CiscoDevNet/yang-explorer](https://github.com/CiscoDevNet/yang-explorer)
  • Web Based GUI
  • More difficult to setup

• **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 -f tree -p yang/standard/ietf/RFC yang/standard/ietf/RFC/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

[...]
pyang Tip – JavaScript Tree Output

- Use `pyang -f jstree -p <path to models> <model.yang> >/tmp/ietf.html`
- Produces collapsible Tree / HTML
Building a Plugin/Application

1. Yang Tools
   - Yang Model
   - Generate APIs

2. Maven Build Tools
   - Generated API Definition
   - “API” OSGI Bundle
   - Create API Bundle
   - Deploy

3. Maven Build Tools
   - Plugin source code
   - “Plugin” OSGI Bundle
   - Create Plugin Bundle
   - Deploy

OpenDaylight as a Platform for Network Programmability
NETCONF
NETCONF
IETF network management protocol

- Distinguishes between configuration and state data
- Multiple configuration datastores (candidate, running, startup)
- Configuration change validation and transactions
- Selective data retrieval via filtering
- Streaming and playback of event 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 data and state data exposed as resources
- How to access the data using REST verbs (GET / PUT / POST/ …)
- How to construct URIs to access the data
- HTTP instead of SSH for transport
- JSON in addition to XML for data encoding

In Summary:

RESTCONF provides lighter-weight interface to network datastores leveraging well known combination of REST and JSON
RESTCONF URI & JSON Example

```
pyang -f tree ietf-interfaces.yang
module: ietf-interfaces
    +--rw interfaces
        | +--rw interface* [name]
        |     +--rw name string
        |     +--rw description?
        .
        .
        .

http://172.16.126.250:8008/api/running/interfaces

"ietf-interfaces:interfaces": {
    "interface": [
        {
            "name": "GigabitEthernet3",
            "description": "To CE-1"
        }
    ]
}
```
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

Manageability Infra
- Config DB

Services
- BGP
- QoS
- VXLAN

Manageability

Infrastructure

Config

DB

APPLICATION

Transport

Any

Network Device

Manageability

Infra

Any (C, Java, Python)

Application

Any (Java, Python, Perl, PHP)

Transport

YANG-based XML

YANG-based XML/JSON

Network Device

Manageability Infra

Config DB

Services

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.
Installation
# Distributions

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

<table>
<thead>
<tr>
<th>Release</th>
<th>Edition</th>
<th>Version</th>
<th>Release date</th>
<th>Downloads</th>
<th>Virtual Machines</th>
<th>Documentation</th>
</tr>
</thead>
</table>
| Carbon    | n/a     | n/a     | May 26, 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 |
| Carbon SR1 | n/a     | n/a     | July 14, 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 |
| Boron-SR3 | n/a     | n/a     | April 6, 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 |
$ unzip distribution-karaf-0.6.1-Carbon.zip
Archive: distribution-karaf-0.6.1-Carbon.zip
creating: distribution-karaf-0.6.1-Carbon ...

$ cd distribution-karaf-0.6.1-Carbon/
$ ./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 3s. Bundle stats: 64 active, 64 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

![Diagram of Karaf with OSGi Framework (Equinox)]
**OpenDaylight User Interface – DLUX**

**opendaylight-user@root>feature:list -i**

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Installed</th>
<th>Repository</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Karaf standard feature</td>
</tr>
<tr>
<td>config</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Provide OSGi ConfigAdmin support</td>
</tr>
<tr>
<td>region</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Provide Region Support</td>
</tr>
<tr>
<td>package</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Package commands and mbeans</td>
</tr>
<tr>
<td>kar</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Provide KAR (KARaf archive) support</td>
</tr>
<tr>
<td>ssh</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Provide a SSHd server on Karaf</td>
</tr>
<tr>
<td>management</td>
<td>3.0.8</td>
<td>x</td>
<td>standard-3.0.8</td>
<td>Provide a JMX MBeanServer and Mbeans</td>
</tr>
</tbody>
</table>

**opendaylight-user@root>feature:list | grep dlux**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl-unimgr-dlux</td>
<td>0.2.1-Carbon</td>
<td>OpenDaylight :: UniMgr :: dlux</td>
</tr>
<tr>
<td>odl-dlux-core</td>
<td>0.5.1-Carbon</td>
<td>OpenDaylight dlux minimal feature</td>
</tr>
<tr>
<td>odl-dluxapps-applications</td>
<td>0.5.1-Carbon</td>
<td>OpenDaylight DluxApps all applications</td>
</tr>
<tr>
<td>odl-dluxapps-nodes</td>
<td>0.5.1-Carbon</td>
<td>Enable nodes in OpenDaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-topology</td>
<td>0.5.1-Carbon</td>
<td>Enable topology in OpenDaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangui</td>
<td>0.5.1-Carbon</td>
<td>Enable Yang UI in OpenDaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangman</td>
<td>0.5.1-Carbon</td>
<td>Enable Yangman in OpenDaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangvisualizer</td>
<td>0.5.1-Carbon</td>
<td>Enable Yang visualizer in OpenDaylight</td>
</tr>
<tr>
<td>odl-dluxapps-yangutils</td>
<td>0.5.1-Carbon</td>
<td>Loads Yangutils library in OpenDaylight</td>
</tr>
</tbody>
</table>

OpenDaylight as a Platform for Network Programmability © 2017 Cisco and/or its affiliates. All rights reserved. Cisco Public
## Install DLUX Feature

```bash
.opendaylight-user@root> feature:install odl-dlux-core odl-dluxapps-applications

.opendaylight-user@root> feature:list -i | grep dlux

<table>
<thead>
<tr>
<th>Feature</th>
<th>Version</th>
<th>Installed</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl-dlux-core</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Opendaylight dlux minimal feature</td>
</tr>
<tr>
<td>odl-dluxapps-applications</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Opendaylight DluxApps all applications</td>
</tr>
<tr>
<td>odl-dluxapps-nodes</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Enable nodes in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-topology</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Enable topology in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangui</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Enable Yang UI in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangman</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Enable Yangman in Opendaylight dlux</td>
</tr>
<tr>
<td>odl-dluxapps-yangvisualizer</td>
<td>0.5.1-Carbon</td>
<td>x</td>
<td>Enable Yang visualizer in Opendaylight</td>
</tr>
</tbody>
</table>
```
http://localhost:8181/index.html#/yangui/index
Example Use Cases
OpenDaylight with Mininet, OVSDB and OpenFlow
Honeycomb/VPP using NETCONF

- VPP is a high-performance software forwarder (see http://www.fd.io)
- Honeycomb provides NETCONF management for VPP
Cisco IOS XR using BGP-LS and PCE-P

- Cisco XRv topology in dCloud
  - dCloud is [http://dcloud.cisco.com](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
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

$ 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

- 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
Additional resources
Open Source Dev Center

Your Source for Open Source at Cisco

https://developer.cisco.com/opensource

- Contributions to open source
- Use in products/solutions
- Community forums, blogs
- Developer Events
  - IETF Hackathons and MEF LSO Hackathons featuring open source implementations of open standards
OpenDaylight Microsite
https://developer.cisco.com/opendaylight
Building Applications on Top of OpenDaylight

**AUTODEV**
Visualize and manage IoT sensors embedded in motor vehicles

**BGP and PCEP Pathman**
Visualize topologies and program MPLS traffic engineering (TE) paths

**BIERMAN**
Visualize and manage BIER network topologies within ODL

**DevNet Sample Apps**
Learn how to use ODL and create your own apps that run on top of it

**OpenFlow Manager**
Visualize OpenFlow (OF) topologies, program OF paths and gather OF stats

**PCE-OpenFlow**
Apply policy-based path computation traffic engineering to OpenFlow networks

**YANG Explorer**
YANG browser and RPC builder application to experiment with YANG models

**In-band OAM (IOAM)**
Add operational info to packet as it traverses a path in network

**VPP vBridge Manager**
Define VPP-based virtual bridge domain(s) for L2 connectivity

**YANGMAN**
Dynamically generated UI forms and native JSON representation based on RESTCONF APIs

**OneM2M Plugins**
Extend the functionality of the oneM2M datastore. Protocol conversion, oneM2M data export are examples

**OneM2M TSDR Plugin**
Export oneM2M data to the OpenDaylight Time Series Data Repository

**Pathman SR**
Visualize topologies and program Segment Routing (SR) paths

**Service Function Chaining**
Create and deploy service chains using the NSH protocol as defined in draft-ietf-sfc-ns

**netACL**
Program and manage Access Control Lists (ACLs) on routers in multi-vendor network
Tutorials and Sandboxes

OpenDaylight Boron SR4 with Apps with 8 Nodes v1

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 Karef
- Scenario 5: Install OpenFlow Manager Application
- Scenario 6: Explore Pathman Segment Routing
- Scenario 7: Explore netACL Application
- Scenario 8: Explore Yangman
Continue Your Education

• Become a DevNet Member: http://cs.co/oss17

• Visit our Open Source Dev Center: https://developer.cisco.com/site/opensource/
Thank you!