OpenDaylight Documentation Documentation

Release Beryllium

OpenDaylight Project

Jul 28, 2017

Contents

Contents:

CHAPTER 1

Getting Started Guide

Introduction

The OpenDaylight project is an open source platform for Software Defined Networking (SDN) that uses open protocols to provide centralized, programmatic control and network device monitoring. Like many other SDN controllers, OpenDaylight supports OpenFlow, as well as offering ready-to-install network solutions as part of its platform.

Much as your operating system provides an interface for the devices that comprise your computer, OpenDaylight provides an interface that allows you to connect network devices quickly and intelligently for optimal network performance.

It's extremely helpful to understand that setting up your networking environment with OpenDaylight is not a single software installation. While your first chronological step is to install OpenDaylight, you install additional functionality packaged as Karaf features to suit your specific needs.

Before walking you through the initial OpenDaylight installation, this guide presents a fuller picture of OpenDaylight's framework and functionality so you understand how to set up your networking environment. The guide then takes you through the installation process.

What's different about OpenDaylight

Major distinctions of OpenDaylight's SDN compared to traditional SDN options are the following:

- A microservices architecture, in which a "microservice" is a particular protocol or service that a user wants to enable within their installation of the OpenDaylight controller, for example:
  - A plugin that provides connectivity to devices via the OpenFlow or BGP protocols
  - An L2-Switch or a service such as Authentication, Authorization, and Accounting (AAA).
- Support for a wide and growing range of network protocols beyond OpenFlow, including SNMP, NETCONF, OVSDB, BGP, PCEP, LISP, and more.
- Support for developing new functionality comprised of additional networking protocols and services.

Note: A thorough understanding of the microservices architecture is important for experienced network developers who want to create new solutions in OpenDaylight. If you are new to networking and OpenDaylight, you most likely won't design solutions, but you should comprehend the microservices concept to understand how OpenDaylight works and how it differs from other SDN programs.

What you'll find in this guide

To set up your environment, you first install OpenDaylight followed by the Apache Karaf features that offer the functionality you require. The OpenDaylight Getting Started Guide covers feature descriptions, OpenDaylight installation procedures, and feature installation.

The Getting Started Guide also includes other helpful information, with the following organization:

1. An overview of OpenDaylight and common use models
2. Who should use this guide?
3. OpenDaylight concepts and tools
4. Explanations of OpenDaylight Apache Karaf features and other features that extend network functionality
5. OpenDaylight Beryllium system requirements and Release Notes
6. OpenDaylight installation instructions
7. Feature tables with installation names and compatibility notes

Overview

OpenDaylight performs the following functions:

- Logically centralizes programmatic control of the physical and virtual devices in your network.
- Controls devices with standard, open protocols.
- Provides higher-level abstractions of its capabilities so experienced network engineers and developers can create new applications to customize network setup and administration.

Common use cases for SDN are as follows:

1. Centralized network monitoring, management, and orchestration
2. Proactive network management and traffic engineering
3. Chaining packets through the different VMs, which is known as service function chaining (SFC). SFC enables Network Functions Virtualization (NFV), which is a network architecture concept that virtualizes entire classes of network node functions into building blocks that may connect, or chain together, to create communication services.
4. Cloud - managing both the virtual overlay and the physical underlay beneath it.

Who should use this guide?

OpenDaylight is for users considering open options in network programming. This guide provides information for the following types of users:

1. Those new to OpenDaylight who want to install it and select the features they need to run their network environment using only the command line and GUI. Such users include:
    (a) Students
    (b) Network administrators and engineers.
2. Network engineers and network application developers who want to use OpenDaylight's REST APIs to manage their network programmatically.
3. Network engineers and network application developers who want to write their own OpenDaylight services and plugins for greater functionality. This group of users needs a significant level of expertise in the following areas, which is beyond the scope of this document:
    (a) The YANG modeling language
    (b) The Model-Driven Service Abstraction Layer (MD-SAL)
    (c) Maven build tool
    (d) Management of the shared data store
    (e) How to handle notifications and/or Remote Procedure Calls (RPCs)
4. Developers who would like to join the OpenDaylight community and contribute code upstream. People in this group design offerings such as applications/services, protocol implementations, and so on, to increase OpenDaylight functionality for the benefit of all end-users.

Note: If you develop code to build new functionality for OpenDaylight and push it upstream (not required), it can become part of the OpenDaylight release. Users can then install the features to implement the solution you've created.

OpenDaylight concepts and tools

In this section we discuss some of the concepts and tools you encounter with basic use of OpenDaylight. The guide walks you through the installation process in a subsequent section, but for now familiarize yourself with the information below.

- To date, OpenDaylight developers have formed more than 50 projects to address ways to extend network functionality. The projects are a formal structure for developers from the community to meet, document release plans, code, and release the functionality they create in an OpenDaylight release.

  The typical OpenDaylight user will not join a project team, but you should know what projects are as we refer to their activities and the functionality they create. The Karaf features to install that functionality often share the project team's name.
- **Apache Karaf** provides a lightweight runtime to install the Karaf features you want to implement and is included in the OpenDaylight platform software. By default, OpenDaylight has no pre-installed features.
- After installing OpenDaylight, you install your selected features using the Karaf console to expand networking capabilities. In the Karaf feature list below are the ones you're most likely to use when creating your network environment.

  As a short example of installing a Karaf feature, OpenDaylight Beryllium offers Application Layer Traffic Optimization (ALTO). The Karaf feature to install ALTO is odl-alto-all. On the Karaf console, the command to install it is:

  feature:install odl-alto-all

- **DLUX** is a web-based interface that OpenDaylight provides for you to manage your network. Its Karaf feature installation name is "odl-dlux-core".
  1. DLUX draws information from OpenDaylight's topology and host databases to display the following information:
     (a) The network
     (b) Flow statistics
     (c) Host locations
  2. To enable the DLUX UI after installing OpenDaylight, run:

     feature:install odl-dlux-core

     on the Karaf console.
- **Network embedded Experience (NeXt)** is a developer toolkit that provides tools to draw network-centric topology UI elements that offer visualizations of the following:
  1. Large complex network topologies
  2. Aggregated network nodes
  3. Traffic/path/tunnel/group visualizations
  4. Different layout algorithms
  5. Map overlays
  6. Preset user-friendly interactions

  NeXt can work with DLUX to build OpenDaylight applications. Check out the NeXt_demo for more information on the interface.
- Model-Driven Service Abstraction Layer (MD-SAL) is the OpenDaylight framework that allows developers to create new Karaf features in the form of services and protocol drivers and connects them to one another. You can think of the MD-SAL as having the following two components:
  1. A shared datastore that maintains the following tree-based structures:
  1. The Config Datastore, which maintains a representation of the desired network state.
  2. The Operational Datastore, which is a representation of the actual network state based on data from the managed network elements.
  2. A message bus that provides a way for the various services and protocol drivers to notify and communicate with one another.
- If you're interacting with OpenDaylight through DLUX or the REST APIs while using the the OpenDaylight interfaces, the microservices architecture allows you to select available services, protocols, and REST APIs.

OpenDaylight Karaf Features

This section provides brief descriptions of the most commonly used Karaf features developed by OpenDaylight project teams. They are presented in alphabetical order. OpenDaylight installation instructions and a feature table that lists installation commands and compatibility follow.

- *ALTO*
- *Border Gateway Protocol (including Link-state Distribution (BGP)*
- *Border Gateway Monitoring Protocol (BMP)*
- *Control and Provisioning of Wireless Access Points (CAPWAP)*
- *Controller Shield*
- *Device Identification and Driver Management (DIDM)*
- *DLUX*
- *Fabric as a Service (FaaS)*
- *Group Based Policy (GBP)*
- *Internet of Things Data Management (IoTDM)*
- *Link Aggregation Control Protocol (LACP)*
- *Location Identifier Separation Protocol (LISP) Flow Mapping Service (LISP)*
- *NEMO*
- *NETCONF*
- *NetIDE*
- *OVSDB-based Network Virtualization Services*
- *OpenFlow Configuration Protocol (OF-CONFIG)*
- *OpenFlow plugin*
- *Path Computation Element Protocol (PCEP)*
- *Secure Network Bootstrapping Interface (SNBi)*
- *Service Function Chaining (SFC)*
- *SNMP Plugin*
- *SNMP4SDN*
- *Source-Group Tag Exchange Protocol (SXP)*
- *Topology Processing Framework*
- *Time Series Data Repository (TSDR)*
- *Unified Secure Channel (USC)*
- *VPN Service*
- *Virtual Tenant Network (VTN)*

AAA

Standards-compliant Authentication, Authorization and Accounting Services. RESTCONF is the most common consumer of AAA, which installs the AAA features automatically. AAA provides:

- Support for persistent data stores
- Federation and SSO with OpenStack Keystone

The Beryllium release of AAA includes experimental support for having the database of users and credentials stored in the cluster-aware MD-SAL datastore.

ALTO

Implements the Application-Layer Traffic Optimization (ALTO) base IETF protocol to provide network information to applications. It defines abstractions and services to enable simplified network views and network services to guide application usage of network resources and includes five services:

1. Network Map Service - Provides batch information to ALTO clients in the forms of ALTO network maps.
2. Cost Map Service - Provides costs between defined groupings.
3. Filtered Map Service - Allows ALTO clients to query an ALTO server on ALTO network maps and/or cost maps based on additional parameters.
4. Endpoint Property Service - Allows ALTO clients to look up properties for individual endpoints.
5. Endpoint Cost Service - Allows an ALTO server to return costs directly amongst endpoints.

Border Gateway Protocol (including Link-state Distribution (BGP)

Is a southbound plugin that provides support for Border Gateway Protocol (including Link-state Distribution) as a source of L3 topology information.

Border Gateway Monitoring Protocol (BMP)

Is a southbound plugin that provides support for BGP Monitoring Protocol as a monitoring station.

Control and Provisioning of Wireless Access Points (CAPWAP)

Enables OpenDaylight to manage CAPWAP-compliant wireless termination point (WTP) network devices. Intelligent applications, e.g., radio planning, can be developed by tapping into the operational states made available via REST APIs of WTP network devices.

Controller Shield

Creates a repository called the Unified-Security Plugin (USecPlugin) to provide controller security information to northbound applications, such as the following:

- Collating the source of different attacks reported in southbound plugins
- Gathering information on suspected controller intrusions and trusted controllers in the network

Information collected at the plugin may also be used to configure firewalls and create IP blacklists for the network.

Device Identification and Driver Management (DIDM)

Provides device-specific functionality, which means that code enabling a feature understands the capability and limitations of the device it runs on. For example, configuring VLANs and adjusting FlowMods are features, and there may be different implementations for different device types. Device-specific functionality is implemented as Device Drivers.

DLUX

Web based OpenDaylight user interface that includes:

- An MD-SAL flow viewer
- Network topology visualizer
- A tool box and YANG model that execute queries and visualize the YANG tree

Fabric as a Service (FaaS)

Creates a common abstraction layer on top of a physical network so northbound APIs or services can be more easily mapped onto the physical network as a concrete device configuration.

Group Based Policy (GBP)

Defines an application-centric policy model for OpenDaylight that separates information about application connectivity requirements from information about the underlying details of the network infrastructure. Provides support for:

- Integration with OpenStack Neutron
- Service Function Chaining
- OFOverlay support for NAT, table offsets

Internet of Things Data Management (IoTDM)

Developing a data-centric middleware to act as a oneM2M-compliant IoT Data Broker (IoTDB) and enable authorized applications to retrieve IoT data uploaded by any device.

Link Aggregation Control Protocol (LACP)

LACP can auto-discover and aggregate multiple links between an OpenDaylight-controlled network and LACP-enabled endpoints or switches.

Location Identifier Separation Protocol (LISP) Flow Mapping Service (LISP)

LISP (RFC6830) enables separation of Endpoint Identity (EID) from Routing Location (RLOC) by defining an overlay in the EID space, which is mapped to the underlying network in the RLOC space.

LISP Mapping Service provides the EID-to-RLOC mapping information, including forwarding policy (load balancing, traffic engineering, and so on) to LISP routers for tunneling and forwarding purposes. The LISP Mapping Service can serve the mapping data to data plane nodes as well as to OpenDaylight applications.

To leverage this service, a northbound API allows OpenDaylight applications and services to define the mappings and policies in the LISP Mapping Service. A southbound LISP plugin enables LISP data plane devices to interact with OpenDaylight via the LISP protocol.

NEMO

Is a Domain Specific Language (DSL) for the abstraction of network models and identification of operation patterns. NEMO enables network users/applications to describe their demands for network resources, services, and logical operations in an intuitive way that can be explained and executed by a language engine.

NETCONF

Offers four features:

- odl-netconf-mdsal: NETCONF Northbound for MD-SAL and applications
- odl-netconf-connector: NETCONF Southbound plugin - configured through the configuration subsystem
- odl-netconf-topology: NETCONF Southbound plugin - configured through the MD-SAL configuration datastore
- odl-restconf: RESTCONF Northbound for MD-SAL and applications

NetIDE

Enables portability and cooperation inside a single network by using a client/server multi-controller architecture. It provides an interoperability layer allowing SDN Applications written for other SDN Controllers to run on OpenDaylight. NetIDE details:

- Architecture follows a client/server model: other SDN controllers represent clients with OpenDaylight acting as the server.
- OpenFlow v1.0/v1.3 is the only southbound protocol supported in this initial release. We are planning for other southbound protocols in later releases.
- The developer documentation contains the protocol specifications required for developing plugins for other client SDN controllers.
- The NetIDE Configuration file contains the configurable elements for the engine.

OVSDB-based Network Virtualization Services

Several services and plugins in OpenDaylight work together to provide simplified integration with the OpenStack Neutron framework. These services enable OpenStack to offload network processing to OpenDaylight while enabling OpenDaylight to provide enhanced network services to OpenStack.

OVSDB Services are at parity with the Neutron Reference Implementation in OpenStack, including support for:

- L2/L3
  - The OpenDaylight Layer-3 Distributed Virtual Router is fully on par with what OpenStack offers and now provides completely decentralized Layer 3 routing for OpenStack. ICMP rules for responding on behalf of the L3 router are fully distributed as well.
  - Full support for distributed Layer-2 switching and distributed IPv4 routing is now available.
- Clustering - Full support for clustering and High Availability (HA) is available in the OpenDaylight Beryllium release. In particular, the OVSDB southbound plugin supports clustering that any application can use, and the Openstack network integration with OpenDaylight (through OVSDB Net-Virt) has full clustering support. While there is no specific limit on cluster size, a 3-node cluster has been tested extensively as part of the Beryllium release.

- Security Groups - Security Group support is available and implemented using OpenFlow rules that provide superior functionality and performance over OpenStack Security Groups, which use IPTables. Security Groups also provide support for ConnTrack with stateful tracking of existing connections. Contract-based Security Groups require OVS v2.5 with contract support.
- Hardware Virtual Tunnel End Point (HW-VTEP) - Full HW-VTEP schema support has been implemented in the OVSDB protocol driver. Support for HW-VTEP via OpenStack through the OVSDB-NetVirt implementation has not yet been provided as we wait for full support of Layer-2 Gateway (L2GW) to be implemented within OpenStack.
- Service Function Chaining
- Open vSwitch southbound support for quality of service and Queue configuration Load Balancer as service (LBaaS) with Distributed Virtual Router, as offered in the Lithium release
- Network Virtualization User interface for DLUX

OpenFlow Configuration Protocol (OF-CONFIG)

Provides a process for an Operation Context containing an OpenFlow Switch that uses OF-CONFIG to communicate with an OpenFlow Configuration Point, enabling remote configuration of OpenFlow datapaths.

OpenFlow plugin

Supports connecting to OpenFlow-enabled network devices via the OpenFlow specification. It currently supports OpenFlow versions 1.0 and 1.3.2.

In addition to support for the core OpenFlow specification, OpenDaylight Beryllium also includes preliminary support for the Table Type Patterns and OF-CONFIG specifications.

Path Computation Element Protocol (PCEP)

Is a southbound plugin that provides support for performing Create, Read, Update, and Delete (CRUD) operations on Multiprotocol Label Switching (MPLS) tunnels in the underlying network.

Secure Network Bootstrapping Interface (SNBi)

Leverages manufacturer-installed IEEE 802.1AR certificates to secure initial communications for a zero-touch approach to bootstrapping using Docker. SNBi devices and controllers automatically do the following:

1. Discover each other, which includes:
   (a) Revealing the physical topology of the network
   (b) Exposing each type of a device
   (c) Assigning the domain for each device
2. Get assigned an IP-address
3. Establish secure IP connectivity

SNBi creates a basic infrastructure to host, run, and lifecycle-manage multiple network functions within a network device, including individual network element services, such as:

- Performance measurement

- Traffic-sniffing functionality
- Traffic transformation functionality

SNBi also provides a Linux side abstraction layer to forward elements as well as enhancements to feature the abstraction and bootstrapping infrastructure. You can also use the device type and domain information to initiate controller federation processes.

Service Function Chaining (SFC)

Provides the ability to define an ordered list of network services (e.g. firewalls, load balancers) that are then "stitched" together in the network to create a service chain. SFC provides the chaining logic and APIs necessary for OpenDaylight to provision a service chain in the network and an end-user application for defining such chains. It includes:

- YANG models to express service function chains
- SFC receiver for Intent expressions from REST & RPC
- UI for service chain construction
- LISP support
- Function grouping for load balancing
- OpenFlow renderer for Network Service Headers, MPLS, and VLAN
- Southbound REST interface
- IP Tables-based classifier for grouping packets into selected service chains
- Integration with OpenDaylight GBP project
- Integration with OpenDaylight OVSDB NetVirt project

SNMP Plugin

The SNMP southbound plugin allows applications acting as an SNMP Manager to interact with devices that support an SNMP agent. The SNMP plugin implements a general SNMP implementation, which differs from the SNMP4SDN as that project leverages only select SNMP features to implement the specific use case of making an SNMP-enabled device emulate some features of an OpenFlow-enabled device.

SNMP4SDN

Provides a southbound SNMP plugin to optimize delivery of SDN controller benefits to traditional/legacy ethernet switches through the SNMP interface. It offers support for flow configuration on ACLs and enables flow configuration via REST API and multi-vendor support.

Source-Group Tag Exchange Protocol (SXP)

Enables creation of a tag that allows you to filter traffic instead of using protocol-specific information like addresses and ports. Via SXP an external entity creates the tags, assigns them to traffic appropriately, and publishes information about the tags to network devices so they can enforce the tags appropriately.

More specifically, SXP Is an IETF-published control protocol designed to propagate the binding between an IP address and a source group, which has a unique source group tag (SGT). Within the SXP protocol, source groups with common network policies are endpoints connecting to the network. SXP updates the firewall with SGTs, enabling the firewalls to create topology-independent Access Control Lists (ACLs) and provide ACL automation.

SXP source groups have the same meaning as endpoint groups in OpenDaylight's Group Based Policy (GBP), which is used to manipulate policy groups, so you can use OpenDaylight GPB with SXP SGTs. The SXP topology-independent policy definition and automation can be extended through OpenDaylight for other services and networking devices.

Topology Processing Framework

Provides a framework for simplified aggregation and topology data query to enable a unified topology view, including multi-protocol, Underlay, and Overlay resources.

Time Series Data Repository (TSDR)

Creates a framework for collecting, storing, querying, and maintaining time series data in OpenDaylight. You can leverage various data-driven applications built on top of TSDR when you install a datastore and at least one collector.

Functionality of TDSR includes:

- Data Query Service - For external data-driven applications to query data from TSDR through REST APIs
- NBI integration with Grafana - Allows visualization of data collected in TSDR using Grafana
- Data Purging Service - Periodically purges data from TSDR
- Data Collection Framework - Data Collection framework to allow plugging in of various types of collectors
- HSQL data store - Replacement of H2 data store to remove third party component dependency from TSDR
- Enhancement of existing data stores including HBase to support new features introduced in Beryllium
- Cassandra data store - Cassandra implementation of TSDR SPIs
- NetFlow data collector - Collect NetFlow data from network elements
- SNMP Data Collector - Integrates with SNMP plugin to bring SNMP data into TSDR
- Syslog data collector - Collects syslog data from network elements

TSDR has multiple features to enable the functionality above. To begin, select one of these data stores:

- odl-tsdr-hsqldb-all
- odl-tsdr-hbase
- odl-tsdr-cassandra

Then select any "collectors" you want to use:

- odl-tsdr-openflow-statistics-collector
- odl-tsdr-netflow-statistics-collector
- odl-tsdr-controller-metrics-collector
- odl-tsdr-snmp-data-collector
- odl-tsdr-syslog-collector

See these TSDR_Directions for more information.

Unified Secure Channel (USC)

Provides a central server to coordinate encrypted communications between endpoints. Its client-side agent informs the controller about its encryption capabilities and can be instructed to encrypt select flows based on business policies.

A possible use case is encrypting controller-to-controller communications; however, the framework is very flexible, and client side software is available for multiple platforms and device types, enabling USC and OpenDaylight to centralize the coordination of encryption across a wide array of endpoint and device types.

VPN Service

Implements the infrastructure services required to support L3 VPN service. It initially leverages open source routing applications as pluggable components. L3 services include:

- The L3 VPN Manager
- MP-BGP Routing Stack
- MPLS Label Manager
- NextHop Manager
- FIB Service & Openstack Neutron Service

The VPN Service offers:

- An API for L3 VPN Services
- Integration with open source routing suites, including Quagga & Ryu
- OpenStack Integration with BGPVPN_Blueprint for end-to-end integration
- OpenStack Neutron integration
- VPN Service upstreamed as part of SDN-distributed routing and the VPN (SDNVPN) project of Open Platform for NFV project (OPNFV) (available in Brahmaputra release)
- Network Overlay solution necessary for a Datacenter/Cloud environment

Virtual Tenant Network (VTN)

Provides multi-tenant virtual network on an SDN controller, allowing you to define the network with a look and feel of a conventional L2/L3 network. Once the network is designed on VTN, it automatically maps into the underlying physical network and is then configured on the individual switch, leveraging the SDN control protocol.

By defining a logical plane with VTN, you can conceal the complexity of the underlying network and better manage network resources to reduce network configuration time and errors.

OpenDaylight Experimental Features

Messaging4Transport

Adds AMQP bindings to the MD-SAL, which makes all MD-SAL APIs available via that mechanism. AMQP bindings integration exposes the MD-SAL datatree, rpcs, and notifications via AMQP, when installed.

Network Intent Composition (NIC)

Offers an interface with an abstraction layer for you to communicate "intentions," i.e., what you expect from the network. The Intent model, which is part of NIC's core architecture, describes your networking services requirements and transforms the details of the desired state to OpenDaylight. NIC has four features:

- odl-nic-core-hazelcast: Provides the following:
  - A distributed intent mapping service implemented using hazelcast, which stores metadata needed to process Intent correctly
  - An intent REST API to external applications for Create, Read, Update, and Delete (CRUD) operations on intents, conflict resolution, and event handling
- odl-nic-core-mdsal: Provides the following:
  - A distributed Intent mapping service implemented using MD-SAL, which stores metadata needed to process Intent correctly
  - An Intent rest API to external applications for CRUD operations on Intents, conflict resolution, and event handling
- odl-nic-console: Provides a Karaf CLI extension for Intent CRUD operations and mapping service operations
- Four renderers to provide specific implementations to render the Intent:
  - Virtual Tenant Network Renderer
  - Group Based Policy Renderer
  - OpenFlow Renderer
  - Network MOdeling Renderer

UNI Manager Plug-in (Unimgr)

Formed to initiate the development of data models and APIs that facilitate OpenDaylight software applications' and/or service orchestrators' ability to configure and provision connectivity services.

YANG-PUBSUB

An experimental feature Plugin that allows subscriptions to be placed on targeted subtrees of YANG datastores residing on remote devices. Changes in YANG objects within the remote subtree can be pushed to OpenDaylight as specified and don't require OpenDaylight to make continuous fetch requests. YANG-PUBSUB is developed as a Java project. Development requires Maven version 3.1.1 or later.

Other features

OpFlex

Provides the OpenDaylight OpFlex Agent , which is a policy agent that works with Open vSwitch (OVS), to enforce network policy, e.g., from Group-Based Policy, for locally-attached virtual machines or containers.

Network embedded Experience (NeXt)

Provides a network-centric topology UI that offers visualizations of the following:

1. Large complex network topologies
2. Aggregated network nodes
3. Traffic/path/tunnel/group visualizations
4. Different layout algorithms
5. Map overlays
6. Preset user-friendly interactions

NeXt can work with DLUX to build OpenDaylight applications. NeXt does not support Internet Explorer. Check out the NeXt_demo for more information on the interface.

API

We are in the process of creating automatically generated API documentation for all of OpenDaylight. The following are links to the preliminary documentation that you can reference. We will continue to add more API documentation as it becomes available.

- mdsal
- odlparent
- yangtools

Installing OpenDaylight

You complete the following steps to install your networking environment, with specific instructions provided in the subsections below.

Before detailing the instructions for these, we address the following: Java Runtime Environment (JRE) and operating system information Target environment Known issues and limitations

Install OpenDaylight

Downloading and installing OpenDaylight

The default distribution can be found on the OpenDaylight software download page: http://www.opendaylight.org/software/downloads

The Karaf distribution has no features enabled by default. However, all of the features are available to be installed.

Note: For compatibility reasons, you cannot enable all the features simultaneously. We try to document known incompatibilities in the *Install the Karaf features* section below.

Running the karaf distribution

To run the Karaf distribution:

1. Unzip the zip file.
2. Navigate to the directory.
3. run `./bin/karaf`.

For Example:

```
$ ls distribution-karaf-0.4.0-Beryllium.zip
distribution-karaf-0.4.0-Beryllium.zip
$ unzip distribution-karaf-0.4.0-Beryllium.zip
Archive:  distribution-karaf-0.4.0-Beryllium.zip
   creating: distribution-karaf-0.4.0-Beryllium/
   creating: distribution-karaf-0.4.0-Beryllium/configuration/
   creating: distribution-karaf-0.4.0-Beryllium/data/
   creating: distribution-karaf-0.4.0-Beryllium/data/tmp/
   creating: distribution-karaf-0.4.0-Beryllium/deploy/
   creating: distribution-karaf-0.4.0-Beryllium/etc/
   creating: distribution-karaf-0.4.0-Beryllium/externalapps/
...
  inflating: distribution-karaf-0.4.0-Beryllium/bin/start.bat
  inflating: distribution-karaf-0.4.0-Beryllium/bin/status.bat
  inflating: distribution-karaf-0.4.0-Beryllium/bin/stop.bat
$ cd distribution-karaf-0.4.0-Beryllium
$ ./bin/karaf

    ________                       ________                .__  .__       .__     __
    \_____  \ ______   ____   ____ \______ \ _____  ___.__.\|  \| \|__\| ____ \|  \|___
↪_/  \|_
     /   \|   \\____ \_/ __ \ /    \ \|    \|  \\__  \<   \|  \|\|  \| \|  \|/ ___\\|␣
↪ \|  \   __\
    /    \|    \  \|_> >  ___/\|   \|  \|    `   \/ __ \\___  \|\|  \|_\|  / /_/  > ␣
↪ Y  \  \|
    \_______  /   __/ \___  >___\|  /_______  (____  / ____\|\|____/__\___  /\|___\| ␣
↪/__\|
            \/\|__\|        \/     \/        \/     \/\/            /_____/      \/
```

- Press `tab` for a list of available commands
- Typing `[cmd] --help` will show help for a specific command.
- Press `ctrl-d` or type `system:shutdown` or `logout` to shutdown OpenDaylight.

Install the Karaf features

To install a feature, use the following command, where feature1 is the feature name listed in the table below:

```
feature:install <feature1>
```

You can install multiple features using the following command:

```
feature:install <feature1> <feature2> ... <featureN-name>
```

Note: For compatibility reasons, you cannot enable all Karaf features simultaneously. The table below documents feature installation names and known incompatibilities.Compatibility values indicate the following:

- **all** - the feature can be run with other features.
- **self+all** - the feature can be installed with other features with a value of **all**, but may interact badly with other features that have a value of **self+all**. Not every combination has been tested.

Uninstalling features

To uninstall a feature, you must shut down OpenDaylight, delete the data directory, and start OpenDaylight up again.

Important: Uninstalling a feature using the Karaf feature:uninstall command is not supported and can cause unexpected and undesirable behavior.

Listing available features

To find the complete list of Karaf features, run the following command:

```
feature:list
```

To list the installed Karaf features, run the following command:

```
feature:list -i
```

Features to implement networking functionality provide release notes, which you can find in the *Project-specific Release Notes* section.

Karaf running on Windows 10

Windows 10 cannot be identify by Karaf (equinox). Issue occurs during installation of karaf features e.g.:

```
opendaylight-user@root>feature:install odl-restconf
Error executing command: Can't install feature odl-restconf/0.0.0:
Could not start bundle mvn:org.fusesource.leveldbjni/leveldbjni-all/1.8-odl in␣
↪feature(s) odl-akka-leveldb-0.7: The bundle "org.fusesource.leveldbjni.leveldbjni-
↪all_1.8.0 [300]" could not be resolved. Reason: No match found for native code:␣
↪META-INF/native/windows32/leveldbjni.dll; processor=x86; osname=Win32, META-INF/
↪native/windows64/leveldbjni.dll; processor=x86-64; osname=Win32, META-INF/native/
↪osx/libleveldbjni.jnilib; processor=x86; osname=macosx, META-INF/native/osx/
↪libleveldbjni.jnilib; processor=x86-64; osname=macosx, META-INF/native/linux32/
↪libleveldbjni.so; processor=x86; osname=Linux, META-INF/native/linux64/
↪libleveldbjni.so; processor=x86-64; osname=Linux, META-INF/native/sunos64/amd64/
↪libleveldbjni.so; processor=x86-64; osname=SunOS, META-INF/native/sunos64/sparcv9/
↪libleveldbjni.so; processor=sparcv9; osname=SunOS
```

Workaround is to add

org.osgi.framework.os.name = Win32

to the karaf file

etc/system.properties

The workaround and further info are in this thread: http://stackoverflow.com/questions/35679852/karaf-exception-is-thrown-while-installing-org-fusesource-leveldbjni

Beryllium features

Feature Name	Feature Description
Authentication	Enables authentication with support for federation using Apache Shiro
BGP	Provides support for Border Gateway Protocol (including Link-State Distribution) as a source
BMP	Provides support for BGP Monitoring Protocol as a monitoring station
DIDM	Device Identification and Driver Management
Centinel	Provides interfaces for streaming analytics
DLUX	Provides an intuitive graphical user interface for OpenDaylight
Fabric as a Service (Faas)	Creates a common abstraction layer on top of a physical network so northbound APIs or serv
Group Based Policy	Enables Endpoint Registry and Policy Repository REST APIs and associated functionality fo
GBP User Interface	Enables a web-based user interface for Group Based Policy
GBP FaaS renderer	Enables the Fabric as a Service renderer for Group Based Policy
GBP Neutron Support	Provides OpenStack Neutron support using Group Based Policy
L2 Switch	Provides L2 (Ethernet) forwarding across connected OpenFlow switches and support for host
LACP	Enables support for the Link Aggregation Control Protocol
LISP Flow Mapping	Enables LISP control plane services including the mapping system services REST API and L
NEMO CLI	Provides intent mappings and implementation with CLI for legacy devices
NEMO OpenFlow	Provides intent mapping and implementation for OpenFlow devices
NetIDE	Enables portabilty and cooperation inside a single network by using a client/server multi-cont
NETCONF over SSH	Provides support to manage NETCONF-enabled devices over SSH
OF-CONFIG	Enables remote configuration of OpenFlow datapaths
OVSDB OpenStack Neutron	OpenStack Network Virtualization using OpenDaylight's OVSDB support
OVSDB Southbound	OVSDB MDSAL southbound plugin for Open_vSwitch schema
OVSDB HWVTEP Southbound	OVSDB MDSAL hwvtep southbound plugin for the hw_vtep schema
OVSDB NetVirt SFC	OVSDB NetVirt support for SFC
OpenFlow Flow Programming	Enables discovery and control of OpenFlow switches and the topoology between them
OpenFlow Table Type Patterns	Allows OpenFlow Table Type Patterns to be manually associated with network elements
Packetcable PCMM	Enables flow-based dynamic QoS management of CMTS use in the DOCSIS infrastructure ar
PCEP	Enables support for PCEP
RESTCONF API Support	Enables REST API access to the MD-SAL including the data store
SDNinterface	Provides support for interaction and sharing of state between (non-clustered) OpenDaylight in
SFC over L2	Supports implementing Service Function Chaining using Layer 2 forwarding
SFC over LISP	Supports implementing Service Function Chaining using LISP
SFC over REST	Supports implementing Service Function Chaining using REST CRUD operations on networl
SFC over VXLAN	Supports implementing Service Function Chaining using VXLAN tunnels
SNMP Plugin	Enables monitoring and control of network elements via SNMP
SNMP4SDN	Enables OpenFlow-like control of network elements via SNMP
SSSD Federated Authentication	Enables support for federated authentication using SSSD
Secure tag eXchange Protocol (SXP)	Enables distribution of shared tags to network devices
Time Series Data Repository (TSDR)	Enables support for storing and querying time series data with the default data collector for O

Feature Name	Feature Description
TSDR Data Collectors	Enables support for various TSDR data sources (collectors) including OpenFlow statistics, N
TSDR Data Stores	Enables support for TSDR data stores including HSQLDB, HBase, and Cassandra
Topology Processing Framework	Enables merged and filtered views of network topologies
Unified Secure Channel (USC)	Enables support for secure, remote connections to network devices
VPN Service	Enables support for OpenStack VPNaaS
VTN Manager	Enables Virtual Tenant Network support
VTN Manager Neutron	Enables OpenStack Neutron support of VTN Manager

Other Beryllium features

Table 1.2: Other Beryllium features

Feature Name	Feature Description	Karaf feature name	Compatibility
OpFlex	Provides OpFlex agent for Open vSwitch to enforce network policy, such as GBP, for locally-attached virtual machines or containers	n/a	all
NeXt	Provides a developer toolkit for designing network-centric topology user interfaces	n/a	all

Experimental Beryllium Features

The following functionality is labeled as experimental in OpenDaylight Beryllium and should be used accordingly. In general, it is not supposed to be used in production unless its limitations are well understood by those deploying it.

Table 1.3: Other Beryllium features

Feature Name	Feature Description	Karaf feature name	Compatibility
Authorization	Enables configurable role-based authorization	odl-aaa-authz	all
ALTO	Enables support for Application-Layer Traffic Optimization	odl-alto-core	self+all
CAPWAP	Enables control of supported wireless APs	odl-capwap-ac-rest	all
Clustered Authentication	Enables the use of the MD-SAL clustered data store for the authentication database	odl-aaa-authn-mdsal-cluster	all
Controller Shield	Provides controller security information to northbound applications	odl-usecplugin	all
GBP IO Visor Renderer	Provides support for rendering Group Based Policy to IO Visor	odl-groupbasedpolicy-iovisor	all
Internet of Things Data Management	Enables support for the oneM2M specification	odl-iotdm-onem2m	all
LISP Flow Mapping OpenStack Network Virtualization	Experimental support for OpenStack Neutron virtualization	odl-lispflowmapping-neutron	self+all
Messaging4Transport	Introduces an AMQP Northbound to MD-SAL	odl-messaging4transport	all
Network Intent Composition (NIC)	Provides abstraction layer for communcating network intents (including a distributed intent mapping service REST API) using either Hazelcast or the MD-SAL as the backing data store for intents	odl-nic-core-hazelcast or odl-nic-core-mdsal	all
NIC Console	Provides a Karaf CLI extension for intent CRUD operations and mapping service operations	odl-nic-console	all
NIC VTN renderer	Virtual Tenant Network renderer for Network Intent Composition	odl-nic-renderer-vtn	self+all
NIC GBP renderer	Group Based Policy renderer for Network Intent Composition	odl-nic-renderer-gbp	self+all
NIC OpenFlow renderer	OpenFlow renderer for Network Intent Composition	odl-nic-renderer-of	self+all
NIC NEMO renderer	NEtwork MOdeling renderer for Network Intent Composition	odl-nic-renderer-nemo	self+all
OVSDB NetVirt UI	OVSDB DLUX UI	odl-ovsdb-ui	all
Secure Networking Bootstrap	Defines a SNBi domain and associated white lists of devices to be accommodated to the domain	odl-snbi-all	self+all
UNI Manager	Initiates the development of data models and APIs to facilitate configuration and provisioning connectivity services for OpenDaylight applications and services	odl-unimgr	all
YANG PUBSUB	Allows subscriptions to be placed on targeted subtrees of YANG datastores residing on remote devices to obviate the need for OpenDaylight to make continuous fetch requests	odl-yangpush-rest	all

Install support for REST APIs

Most components that offer REST APIs will automatically load the RESTCONF API Support component, but if for whatever reason they seem to be missing, install the “odl-restconf” feature to activate this support.

Release Notes

Target Environment

For Execution

The OpenDaylight Karaf container, OSGi bundles, and Java class files are portable and should run on any Java 7- or Java 8-compliant JVM to run. Certain projects and certain features of some projects may have additional requirements. Those are noted in the project-specific release notes.

Projects and features which have known additional requirements are:

- TCP-MD5 requires 64-bit Linux
- TSDR has extended requirements for external databases
- Persistence has extended requirements for external databases
- SFC requires addition features for certain configurations
- SXP depends on TCP-MD5 on thus requires 64-bit Linux
- SNBI has requirements for Linux and Docker
- OpFlex requires Linux
- DLUX requires a modern web browser to view the UI
- AAA when using federation has additional requirements for external tools
- VTN has components which require Linux

For Development

OpenDaylight is written primarily in Java project and primarily uses Maven as a build tool Consequently the two main requirements to develop projects within OpenDaylight are:

- A Java 8-compliant JDK
- Maven 3.1.1 or later

Applications and tools built on top of OpenDaylight using it's REST APIs should have no special requirements beyond whatever is needed to run the application or tool and make the REST calls.

In some places, OpenDaylight makes use of the Xtend language. While Maven will download the appropriate tools to build this, additional plugins may be required for IDE support.

The projects with additional requirements for execution typically have similar or more extensive additional requirements for development. See the project-specific release notes for details.

Known Issues and Limitations

Other than as noted in project-specific release notes, we know of the following limitations:

- Migration from Helium, Lithium and Beryllium to Boron has not been extensively tested. The per-project release notes include migration and compatibility information when it is known.
- There are scales beyond which the controller has been unreliable when collecting flow statistics from OpenFlow switches. In tests, these issues became apparent when managing thousands of OpenFlow switches, however this may vary depending on deployment and use cases.

Security Advisories

All OpenDaylight Security Advisories can be found on the Security Advisories wiki page. Of particular note to OpenDaylight Beryllium users are:

- CVE-2017-1000359
- CVE-2017-1000360
- CVE-2017-1000357
- CVE-2017-1000358
- CVE-2017-1000361

There are known and documented mitigations described on the Security Advisory page linked above. Because of the efficacy of the mitigations, we do not intend to release another version of Beryllium to address them. Instead, we encourage all of those who are using Beryllium to carefully understand the mitigations in the context of their deployments.

Project-specific Release Notes

For the release notes of individual projects, please see the following pages on the OpenDaylight Wiki.

TBD: add Boron release notes

- Authentication, Authorization and Accounting (AAA)
- ALTO
- BGPCEP
- Controller
- Control And Provisioning of Wireless Access Points (CAPWAP)
- Identification and Driver Management (DIDM)
- DLUX
- FaaS
- Group_Based_Policy (GPB)
- Internet of Things Data Management (IoTDM)
- L2_Switch
- Link Aggregation Control Protocol (LACP)
- LISP_Flow_Mapping
- MDSAL
- NEMO
- NETCONF
- NetIDE
- NeXt
- Network Intent Composition (NIC)
- Neutron_Northbound
- OF-Config

- OpFlex
- OpenFlow_Plugin
- OpenFlow_Protocol_Library
- OVSDB_Netvirt
- Packet_Cable / PCMM
- SDN_Interface_Application
- Secure Network Bootstrapping Infrastructure (SNBI)
- SNMP4SDN
- SNMP_Plugin
- Secure tag eXchange Protocol (SXP)
- Service Function Chaining (SFC)
- TCPMD5
- Time Series Data Repository (TSDR)
- Table Type Patterns (TTP)
- Topology_Processing_Framework
- Unified Secure Channel (USC)
- VPN_Service
- Virtual Tenant Network (VTN)
- YANG_Tools

Projects without Release Notes

The following projects participated in Boron, but intentionally do not have release notes.

- **Documentation Project** produced this and the other downloadable documentation
- **Integration Group** hosted the OpenDaylight-wide tests and main release distribution
- **Release Engineering - autorelease** was used to build the Boron release artifacts and including the main release download.

Beryllium-SR3 Release Notes

This page details changes and bug fixes between the Beryllium Stability Release 2 (Beryllium-SR2) and the Beryllium Stability Release 3 (Beryllium-SR3) of OpenDaylight.

Projects with No Noteworthy Changes

The following projects had no noteworthy changes in the Beryllium-SR3 Release:

- ALTO
- Centinel
- Control And Provisioning of Wireless Access Points (CAPWAP)

- Controller Shield
- Device Identification and Driver Management (DIDM)
- Messaging4Transport
- NEtwork MOdeling (NEMO)
- NeXt UI Toolkit
- Network Intent Composition (NIC)
- Neutron Northbound
- Packet Cable/PCMM
- SDN Interface Application (SDNi)
- SNMP Plugin
- Service Function Chaining
- YANG PUBSUB

Authentication, Authorization and Accounting (AAA)

- ed72f9 Fix for BUG-6082 - idpmapping will failed for the case sensitivity
- ee6b8d Modify idmtool insecure option to work with older versions of requests
- 77d2cb Enhance idmtool to allow disabling https certificate verification
- 935bab Git ignore .checkstyle file create by Eclipse Checkstyle plugin

BGP PCEP

- 72cba2 BUG-6264: Beryllium SR3 Build Unstable - when waiting for expected number of messages, try every 50 msecs till a max of 10 secs
- c2572b BUG-6120: Fix intermitent test fail
- b07d1c BUG-5742: Race condition when creating Application Peer on clustering
- d12fd5 BUG-5610: PCRpt message with vendor specific object throws exception
- a2c7a4 BUG-6108: Fix IAE on ApplicationPeer
- 5f7b28 BUG-6084: get restart time from open message error - fix size of left-shift while calculating graceful restart capability restart time
- 7b9026 Fix failing unit tests
- 59f858 Fix unit test regression after netty version bump
- 95768e removed precondition checks for v4/v6 next-hops for v4/v6 routes
- d14d8d BUG-6005: PCErr generated while parsing of received PCRpt message is not sent out - use Channel#writeAndFlush instead of ChannelHandlerContext#write when sending out PCEP error message so that decode handler is invoked - added listener to ChannelFuture to log result of send operation
- 694404 BUG-6019: Wrong path name for route distinguisher
- dc7453 BUG-6001: Injecting route with missing next-hop value causes exception in reachability topology builder - added check in reachability topology builder to handle scenario when next-hop value is null - entry will be skipped from topology processing in such cases

- f18e4f BUG-5978: Unrecognized attribute flagged Well Known - set optional bit when serializing unrecognized attributes - updated unit-test
- cbb0ca BUG-5763 Disallow redelegation for PCC-initiated LSP
- b1550e BUG-5548: NH serializer removal
- ad543a BUG-5548: Wrong NH handler picked up
- 7840f5 Remove nexusproxy property as it is inherited via odlparent
- 996f5f BUG-5855: Transitive Unrecognized Attribute not transiting - added missing serializer for BGP unrecognized attributes - added unit-test to test the unrecognized attributes serializer
- 413133 Remove eclipse project files, add more extensions to gitignore
- 486a89 BUG-5731: Send Error Message if LSP-IDENTIFIERS TLV is missing
- 205e54 BUG-5689: Unhandled message is causing failure
- 3cfdd3 BUG-5612: ODL(PCEP) infinitely waits for the response from peer for addlsp (cherry-pick) - added configurable timeout (default 30 secs) while processing RPC requests which need response from PCC - updated unit-test

Controller

- d8b664 Change default value of parameter “auto-down-unreachable-after”
- ca64cf Reduce ConflictingVersionException log level to debug
- bac013 Change count type in the cars model
- 302fba Force install snapshot when follower log is ahead
- b75958 Clear leaderId when election timeout occurs in non-voting follower
- b8e210 Add ServerConfigPayload to InstallSnapshot message
- 9583d2 Backport InstallSnapshot message serialization changes
- 7a53dd Add option to enable/disable basic DCL and/or DTCL
- 420761 Fix intermittent unit test failures
- 3066f5 BUG-6106: Prevent flood of quarantine messages
- 49ffbb Fix intermittent test failures in ClusterAdminRpcServiceTest
- 0f0acc Fix intermittent test failures in PartitionedLeadersElectionScenarioTest
- 28ecb3 Add voting state to shard mbean FollowerInfo
- 6ce895 Implement cluster admin RPCs to change member voting states
- 2277d0 BUG-5913: Fix ISE in DefaultShardDataChangeListenerPublisher
- 716528 Fix test failures in RaftActorServerConfigurationSupportTest
- 635721 Implement change to voting with no leader
- 248a20 Implement ChangeServersVotingStatus message in RaftActor
- d6b79e Add a few toString() methods
- 4f490f BUG-5504: Fix IllegalStateException handling from commit
- 9eaa34 Debug logging in AbstractLeader is too chatty

- f9fc8c Remove snapshot after startup and fix related bug
- f896f5 Guard against duplicate log indexes
- 01c5a7 Add more debug output in AbstractLeader and Follower
- 66eee2 Update version enforcement to Java 7
- 48782a BUG-5414 introduce EOS inJeopardy flag
- 24ecd4 Make Karaf dump heap on OOM by default

DLUX

- 4cf653 Remove unused property

Documentation

- 4c8edd Fix minor typo in OpenFlow Plugin dev docs
- 0a914c Update VTN Manager adoc for Beryllium
- 9ab283 ADDED UniMgr developer guide
- c0d025 Documentation outline for of-config.
- 65f148 Adding more documentation topics to nic
- 7ee3da Topoprocessing - another minor fixes
- 1de195 Minor fixes
- 19e882 Adding OpenFlow Plugin Lithium config subsystem docs
- cadd5b Print location of built docs
- 986636 Update ALTO Docs for Beryllium
- 2b3b83 Draft of topoprocessing documentation
- 9ca71b Start documentation of topology processing framework
- 649c58 Use at least 4.11 version of bootstrap theme
- 6cf489 Adding OpenFlow Plugin yang model docs
- d2274b Add ci-requirements.txt
- 1ab3ba Force update sphinxbootstrap theme in rtd
- 057ad4 Force update additional dependencies
- 16e6d6 Update to follow style guide
- 5592d2 Fix mobile site logo size too large
- d88f8a Added section on ovsdb-library-developer.adoc.
- 8e7057 Update netconf user guide
- af0b83 changing version from 0.3.0 to Beryllium
- 1bcd17 Documentation outline for centinel.
- fa0cd8 adding .tox to .gitignore
- e9a5f4 Usecplugin Documents upload

- f5c63a Switch docs to use a bootstrap theme
- df35b0 Added known issues to getting started guide - installation
- 0ecd9b Add cluster configuration scripts
- 942515 Make generated html appear in _build/html
- 083f3e make html prettier for robot libdocs
- 94975f Removing the AsciiDoc Getting Started Guide as it's all in reST
- ee3062 Migrating security considerations to reST
- 0be6ad Migrating DLUX, Clustering, Version and XSQL to reST
- 2a1e4d Migrate general installation instructions to reST
- 2dbeaf Migrating project-specific install guides from adoc to reST
- eb590b Migrating release notes from AsciiDoc to reST
- 6019cd Add docs-linkchecker to validate external links
- e18c3a Convert be-exp-features-?.png to list-table
- 80ec96 Convert be-nonfeatures.png to list-table
- 99bcb3 Convert be-features-3.png to list-table
- a4f5d7 Convert be-features-2.png to list-table
- c95290 Convert be-features-1.png to list-table
- a85282 Add support to make docs build using tox
- adf26e Restructure to docs folder for RTD
- ae24a7 Convert Getting Started Guide to Sphinx
- e92697 Initial ReadTheDocs generation
- 876b6a fixing case file flow_filter_example.png in user guide
- a44758 adding .DS_Store to .gitignore
- f22e29 Fix indentation in root pom file.
- 20e4d9 Adding the feature table from Denise's GSG
- 2f99f8 Some last-minute cleanups for Beryllium docs
- 2a4b43 Modified documents with new references for Be release
- ffde28 Fixing mixed case issues with images in VTN docs
- 45b561 Added Security related paragraph for TSDR project
- 8b9f1a VTN Beryllium - Adoc changes.
- 5ce821 Adding Beryllium-SR2 Release Notes
- afacea Update current branch to stable/beryllium
- e2545d Use https to connect to ODL Nexus
- 8ffee7 OVSDB User Guide documentation

Fabric As A Service (FaaS)

- a1ce82 Use utils.mdsal-openflow dependency from netvirt project

Group Based Policy (GBP)

- 120be6 Tests for neutron-ovsdb
- 2d3fa2 Unit tests for ofoverlay
- 228238 Code coverage increasing for faas-renderer
- c551fa Tests for Iovisor root package
- 804531 Test coverage increasing for iovisor.restclient
- 996ba3 Tests coverage increasing for iovisor.endpoint
- 71ce73 Tests for iovisor.sf
- e26053 GBP tests: fixes for unnecessary mocking, method naming and other improvements
- cf601d New tests for listeners
- 76c742 OFoverlay statistics test improvement
- 6eacd8 GBP ofoverlay.sf test improvements

Integration/Distribution

- fffd4b BUG-4296: Add config subsystem file with versions
- ed716c Remove VPN Service from distribution check
- febce1 Change alto feature to not-compatible list

Internet of Things Data Management (IoTDM)

- 596ed9 Changed plugin versions to match this 0.1.3 release
- a8b54f Fix the bug when rcn=4, RSC=400 but still return the content.
- d50cde Change exptime "FOREVER" to "29991231T111111" Change rcn=5, return resourceName, not latest. Fix the bug when create resource with "creator",it will throw RPC error. Remove the commented code.

L2 Switch

- d264e0 Use https to connect to ODL Nexus
- 46b674 BUG-5854: L2Switch stable/berylliumdistribution job does not work correctly Similar BUG-5559 seen for l2switch master. This patch is cherrypick of the bug fix 5559 - replaced hard-coded versions with properties in parent pom - removed/fixed redundant versions in pom - when karaf was run from distribution job, it was throwing exceptions. Fixed root-cause which was that karaf.version was set to old version. Now letting this version flow from the parent - verified using mininet that l2switch features are working as expected when launched from distribution job
- 6e7ddd BUG-5816 - Expired hosts never comeback after timing out
- 2d6ace BUG-5012 - adding configurable support for timing out hosts (in interval of seconds)

- a9b7f7 BUG-2501 Reactive mode -default L2Switch timeout values
- a0ee48 BUG-5629 - Cleaning up Yang files Unused imports are being removed from yang files.

LISP Flow Mapping

- f2ca5b BUG-6024: Fix IPv6 datagram sending
- 864915 Inherit nexusproxy from odlparent
- 9f0737 BUG-5795: Fix SourceDestKey type check

Link Aggregation Control Protocol (LACP)

- bf58c1 Cherry picked changes from https://git.opendaylight.org/gerrit/#/c/39522/ to the beryllium branch.
- dd86f2 Remove unused properties
- 66acb8 Modified LacpNodeExtn.java
- ddbef8 Modified pom.xml and LacpFlow.java

MD-SAL

- 7d64be BUG-6112 - UnionTypeCodec fails to non-identityref value
- 6c7cb5 Optimize UnionTypeCodec
- 82bb64 Enforce non-null class
- b38ccf Optimize BitsCodec
- 89570f BUG-6006 - UnionTypeCodec fails to handle indentityref
- 142b29 LazyDataObject bindingEquals fix
- c882a2 BUG-5970: do not add value to union hashCode/equals/toString
- 4204d5 BUG-5883 - no constructor for indentityref in union
- 030f7d BUG-5882: Wrong placement of deprecated annotation
- e1309c BUG-5845: can not transform ba to bi, when keys contain boolean type
- 6d2a80 BUG-5461: Augmenting a choice without a case from another module causes NPE
- ed8cc4 BUG-5788: enum used as a key does not work
- 073183 BUG-5446: toString() throws exception for ‘type binary’ binding
- aabe7b BUG-4760: YANG leaf named ‘class’ breaks write with netconf connector
- 253697 BUG-1862: Incorrect type transformation in TypeProviderImpl

NETCONF

- c049f3 BUG-3959 - support netconf notification
- 7ac4a1 BUG-6037 - Check if delete request was successful
- d8f8aa Remove obsolete FIXMEs
- 585535 BUG-5909 - PATCH does not report 409 on OptimisticLockFailedException

- d3332c BUG-5897 - PATCH merge operation does nothing
- 204f61 BUG-5915 - PATCH with “target”:”/” error
- cb084f BUG-5509 - HTTP Patch in Restconf doesn’t support general absolute or relative target xpath
- ccbba4 BUG-5509 - HTTP Patch in Restconf doesn’t support general absolute or relative target xpath
- 3f0395 BUG-5898 - PATCH success “ok” field has wrong JSON value
- 035daf Improve logging in netconf test tool
- 10f1f7 BUG-5730 - Delete subset of list items using PATCH?

NetIDE

- 10aa9e Remove unused property

Network Virtualization

- 601acd Move old netvirt files into openstack dir
- 95f9aa BUG-6066 - Improve the logging.
- b44bc6 BUG-6081-ERROR Log Observations - CSIT (SG)
- 61d694 Reduce IT logging
- 7eb90d Fix IT failure when not running DockerOvs docker
- 868d4e BUG-6066 - Improve the logging.
- 8e8254 Reduce logging for openflow and config warnings
- 5ebe4d BUG-6066 - Improve the logging.
- e65f79 BUG-5860 Fix port event wrong ordering
- 816462 Support for IT ping feature
- 3a997b Clean up pom files
- 2a2025 BUG-6017 java.lang.NullPointerException at org.opendaylight.ovsdb.openstack.netvirt.impl.BridgeConfigurationManage createBridges(BridgeConfigurationManagerImpl.java:407)
- 015e39 BUG-6021 java.lang.NullPointerException at org.opendaylight.ovsdb.openstack.netvirt.impl.NeutronL3Adapter.storeNetworkInCleanupCache(NeutronL3Adapter.java:1564).
- 52e70a BUG-6066 Added log message for tunnelports creation.
- 45959e BUG-6070 dpid changes as ports added to bridge
- f037ac Add revision for acl.yang now that it is included in mdsal
- c93994 Use <> Java 7 operator
- c0f384 BUG-6056 - Wrong logging in NetvirtSfcStandaloneOF13Provider
- eee237 Simplify boolean expressions
- a72e09 Mechanical clean-up: semicolons, default access
- 6b4759 BUG-6014 - Named Thread pool Executors for better debugging
- 38af11 BUG-5989: Thrown nullpointerexception while updating the port.

- a86918 BUG-2714 OVSDB needs to be more proactive in reporting errors with underlying OVS instances.
- 77b6f0 BUG-5988: throws NullPointerException while creating a network without br-int interface. * Handled an exception properly wherever throws NullPointerException while creating a network without br-int interface.
- 39f5d8 BUG-2714 OVSDB needs to be more proactive in reporting errors with underlying OVS instances.
- 11fd6a Enable NetVirt Maven site
- 7f3cbf BUG-5894 NullPointerException while deleting the interface from router. * While deleting the interface from router, checking the floatingIp's port uuid is null else delete the respective floating Ip.
- d093d8 Use DockerOvs + test connect to 2 OVSs
- e75b42 BUG-5939 - Communications through external gateway not working
- c18d93 use the right hop for more than one sf
- 716d4f Added support for enable/disable security on a port dynamically.
- 45c5f8 Added support in neutron and it utils for SGs
- 2e01de BUG-5813: Vxlan ports should not be removed in table 110 flow entry unless last VM instance removed from the openstack node. * Before deleting the Vxlan port in flow entry it should check whether the deleted vm instance is last or not. If it is the last vm instance Vxlan port should be delete from source node in flow entry else vxlan port shouldn't be delete.
- 0c34a9 BUG-5614: Ovsdb should not flood the packets to compute nodes unless tenant network exists in the compute node * Before adding tunnel rules, checking the network present or not in src and dst node. If network present in both nodes adding the Vxlan port in flood entry in src and dst.Else do not add vxlan ports.
- b80dc1 Remove ovsdb related in resources
- d8aba5 postman: use 1 for netvirt table offset
- 08e1d0 Added isPortSecurityEnabled check to enable/disable SG.

ODL Root Parent

- 461693 Quick (-Pq) should skip running tests, but not building them
- 68e5f6 Use CustomBundleURLStreamHandlerFactory from features-test
- e1af02 Use more specific dependencies than karaf-maven-plugin
- cbaf17 Introduce "mvn -Pq install" to just build JAR, but no tests, QA etc.
- 7e1d3a Upgrade Netty 4.0.33.Final -> 4.0.37.Final
- 1b688e BUG-4692: Add Netty's native epoll transport

OVSDB Integration

- 4809f5 BUG-6304 - Regression in Connection reconciliation functionality
- bda8b6 BUG-6284 - Cancellation exceptions during operation execution
- 4a820e Fix more Sonar (soon Checkstyle) constant name
- f79ef1 Checkstyle fixes (to be enforced)
- cd7b87 Add support for setting the db schema version
- 93faa9 Checkstyle fixes for ovsdb test dirs

- 955ffe BUG-6185 - southbound system tests failing when run with all other compatible OpenDaylight features.
- 4e9eac Reduce IT logging
- 4fb64d checkstyle fixes for library
- 3ed72f Use StandardCharsets instead of Guava
- 70310b Fix docker compose ps command
- 79f9f8 Fix IT failure when not running DockerOvs docker
- 19cde2 Fixed check-style error caused by odl parents check-style patch
- 8eec85 BUG-5938 - Updated the logging
- a9e17f Fix potential future problems re. hidden fields
- 2c7e99 Checkstyle clean-up src/test/java of southbound-impl
- 01c115 Fix Sonar (soon Checkstyle) constant name
- a015ca BUG-6130 - process only qos and queue creates and updates
- 1f5786 Fix Sonar (soon Checkstyle) TrailingComment
- 95f9e6 Fix Checkstyle "Utility classes should not have (visible) constructor"
- 11e20f Fix ALOTOF Checkstyle violation, and switch over to enforcement.
- 36174f hwvtep yang changes for 1.4.1
- 36eb90 Schema changes for 1.7.0 support
- feeb46 BUG-5938 - Improve the logging.
- ab4037 Checkstyle clean-up invalid license headers containing Authors.
- 5356fc BUG-5938 Updated Log messages.
- 41ee2a BUG-5938 - Updated the logging.
- 07876c Add list of contributors
- 3e1d5d BUG-4790 - 'ifindex' column from Interface table on OVS
- b9e5a6 Use logger instead of System.out.println (found by Checkstyle)
- 846317 ovsdb enable checkstyle on error
- 32588e BUG-5746 - Ovsdb QoS and Queue model enhancements
- f3db50 Fixing sonar bug 1)Add a private constructor to hide the implicit public one. 2)removing the block of commented-out lines as part of code cleanup
- b51e87 BUG-6136 - Southbound plugin throws IllegalStateException when add bridge config via POST. * Checking isPresent() while reading a OvsdbBridgeAugumentation value.
- 6610b0 BUG-5945: Tunnel updates through Genius not working
- 45ff90 Remove verbose logging in TransactUtils
- 26e8bd Patch set 2 ———— 1. Break this up into 2 patches, as Sam suggested. This patch will add new schema dependencies and fix ovsdb code to handle ovsdb node when both schemas are present.
- 918448 BUG-5938 Added log message for OvsdbDataTreeChangeListener.
- 38d337 BUG-5721 - br-int not created in clustered setup
- b02a48 Support for IT Ping feature

- 3b76c4 Make SchemaVersionMismatchException logging consistent
- 9a022b removed unused imports in HwvtepSouthboundUtil.java, ‘org.opendaylight.controller.md.sal.common.api.data.Transactio
  and org.opendaylight.yang.gen.v1.urn.opendaylight.params.xml.ns.yang.ovsdb.hwvtep.rev150901.HwvtepLogicalSwitchAttribut
- a03490 BUG-5976 - Thread leak when connecting/disconnecting ovs nodes
- 250987 Remove OvsdbDataChangeListener
- 941971 BUG-5992 - GBP + SFC integration is broken
- abc18d Add IT for port del dtcl
- 80f927 Removed unused import ‘org.opendaylight.ovsdb.lib.impl.OvsdbConnectionService’.” from HwvtepSouthboundProvider.java
- 1229be Replace wildcard imports
- 7f0ec7 Add ability to create VxLAN-GPE tunnels
- 0ff319 BUG-5944 - Looping INFO messages for UNSUPPORTED AutoAttach OVS
- 947c2a System property awareness in DockerOvs
- 5d95a3 removed unused import ‘org.opendaylight.yang.gen.v1.urn.ietf.params.xml.ns.yang.ietf.inet.types.rev100924.Ipv4Address’ in HwvtepSouthboundMapper.java
- 9d8036 BUG-5479: HWVtep Southbound doesn’t retry connection
- 498c94 BUG-5769
- 6259ad BUG-5506: OVSDB server doesn’t close connection after peer is power down
- e68f75 BUG-5885 - OVSDB plugin failure to update passive ovsdb nodes
- aca1d4 Infra for running ovs dockers
- 202672 Minor sonar listed bug fixed for coding style in HwvtepConnectionManager.java
- 504431 Patch - https://git.opendaylight.org/gerrit/#/c/39061/ is broken .This patch fixes the breakage
- 666862 BUG-5177 - bridge not created if it’s configured northbound while ovs node is disconnected
- 24525e Remove logging entire exception to avoid a noisy stack trace
- 5b690e BUG-5737: enable OVSDB Maven site
- 5aa4a9 Remove redudnant calls for bridge updates
- ea37b5 Get bridge details for delete when using dataTree
- d8e98c BUG-5876 - OVSDB library: Retry when SSL handshake doesn’t begin yet
- c8c4f0 Add MdsalUtilsAsync to make transactions asyncrhonous
- ae7741 Remove the remaining netvirt code from ovsdb
- d37afe Fix Iid for locator-refs in HwvtepTunnelUpdate
- a1e800 BUG-5876 - OVSDB library: Retry when SSL handshake doesn’t begin yet
- 802491 Fix Puppet install in OVSDB Vagrantfile
- af097f Use DataTreeChangeListener instead of DataChangeListener
- 0285a8 Move TpId allocation out of loop
- 6a9285 Fixed hard-coded port 12345 in TestClient
- 920bbd BUG5764: Hwvtep tunnel update/delete not reflected correctly

OpenFlow Configuration Protocol (OF-CONFIG)

- 5828c8 BUG-5780 - can not generate ofconfig topology node

OpenFlow Plugin

- 1126d3 BUG-5390 - Adding Ipv6 L3ArbitraryBitMask feature support.
- bf234d BUG-5390 - Adding Ipv6 L3ArbitraryBitMask feature support.
- 6222ea BUG-6073: Wait for completion of add-group RPCs as needed.
- 92670c Create empty match only once
- eae9e6 Removing duplicate dependencies in pom
- 3836c2 BUG-4148: Fixed the log level.
- e4db06 BUG-6193 - Change in length of DatapathId of a switch
- ac4d6f BUG-6109: Having a null match field is totally fine.
- 88d38b Use odlparent version of maven-dependency-plugin
- 4ba00e BUG-6145 Flows rm'ed from DeviceFlowRegistry after being updated
- d14ca5 Added createEmptyRpcBatchResultFuture method
- 74b243 BUG-5578 Improve FlatBatch service (cleaning)
- 41ecd5 BUG-5578 Improve FlatBatch service (barriers)
- 3ec690 BUG-5590 - writeToTransaction+submitTransaction - New interface TxFacade, DeviceContext now implements this interface - Changed constructor of SalTableServiceImpl to accept TxFacade and NodeId - Changed MultipartRequestOnTheFlyCallback to not need the DeviceContext and accept TxFacade - Updated some methods in StatisticGatheringUtils to not depend on DeviceContext (because of change above) - Updated usages of these classes according to changes. - Updated tests - Fixed submitTransaction in MultipartRequestOnTheFlyCallback endCollecting method
- a6e0a4 BUG-5574: added flat-bulk service registration
- 9815a7 BUG-5574: flat batch proposal
- 83f276 BUG-5574: bulk meters proposal
- 850a11 BUG-5574: introduces sal-groups-batch service impl
- 91d7ac BUG-5574: sal-flows-batch proposal
- cbae6d BUG-6060 failed to push master role request occasionally
- c83bfd BUG-5924 - Reuse Threads using ThreadPool in SystemNotificationListenerImpl
- 334abf BUG-6124 - onIdleSwitchEvent test fails
- fab9f2 BUG-5925 - Reuse Threads using ThreadPool in ConnectionManagerImpl
- 9ca68c BUG-5921 EOS inJeopardy flag
- 12366f BUG-5987 - Operational Inventory data persists in datastore
- 9bb709 BUG-5595 - RpcContext nodeId+xidSequencer (remove dependency on DeviceContext) - Changed RpcContextImpl to not depend on DeviceContext. - Moved isStatisticsRpcEnabled to DeviceManager and DeviceContext - Updated RpcContextManagerImpl to use new RpcContextImpl constructor - Updated tests

- 039e6f BUG-5888: moving the reconciliation process into a different thread to prevent holding on to DTCL thread while provisioning bulk flows/groups.
- 1c8441 BUG-5916: He plugin: Wake up statistics collector thread if RPC fails.
- 88f78b Remove nexusproxy as it is inherited from odlparent
- 1d66f6 Suppressed WARN log of forwardingrules-manager
- 336749 BUG-6085 - Modifying config flow does not work if first flow creation failed
- 1cf810 BUG-6064: switching updates off as on a node updated we need not trigger reconciliation
- c5f8bd BUG-5974: He plugin: Don't invalidate session context that is not valid.
- c756e9 Fixed debug log messages.
- f39d5d BUG-5888 - Changing FRM from clustered DCN to clustered DTCN
- 1c9281 BUG-5953: Fixed NPE in FlowRemovedTranslator.
- 87b7ac Correcting test-failures during openflowplugin-periodic builds
- 99b4d1 BUG-5888: Registering entity-ownership listerner globally
- d13d3e BUG-5636: making table features configurable for the He plugin. DEFAULT will be ON, can be TURNED OFF.
- 3a666e BUG-5839 - Removing stale-marked groups/meters durng reconciliation [cherry-pick from master]
- cdcfe9 BUG-5914 - Flow statistics don't show up on the same flow id, if flow uses IPv6 match with subnet mask
- d96bdb BUG-5841: Enhances bulk-o-matic to stress test Data Store and Openflowplugin RPC Added asciidoc for the same
- 0adac8 added table features skip flag
- d3f498 Add custom compare for ArpMatch objects
- 6f3b6d BUG-4099: Group pointing to ports during reconciliation will be provisioned only after the ports come up after configurable number of retries

OpenFlow Protocol Library

- d7e719 Remove unused property
- 24de99 Fixed netty & checkstyle failures
- 03a603 Scenarios can be in XML file
- c54af8 Utility to test device connections
- d39744 Utility to test device connections

SNMP4SDN

- 299043 Use https to connect ODL Nexus

Secure Network Bootstrapping Infrastructure (SNBI)

- e6e7d4 Remove nexusproxy as it is inherited from odlparent
- 655937 Do not deploy the karaf artifact

Secure tag eXchange Protocol (SXP)

- daac8f BUG-6190 - Export task is not send to remote Peer due to partitioning error
- b975d9 BUG-5975 - SXP may leave opened ChannelHandlerContext for Both mode

TCP-MD5

- 156f96 Remove redundant javadoc/sources configuration

Table Type Patterns (TTP)

- b036c6 Enable Maven Site generation
- 76c82f Fix javadoc validation error in TTPUtils
- 76f9c7 Fix javadoc validation error in TTPYangModelTest
- 054ce7 removing superfluous groupId

Time Series Data Repository (TSDR)

- 8ee8b5 Fix javadoc validation error
- 811be1 Remove nexusproxy as it is inherited from odlparent

Topology Processing Framework

- 7b1c47 Bugfix for BUG-6132
- 0f5b68 Bugfix for BUG-6096
- 4a8654 Enable Maven Site generation
- 4ca4e1 Fix javadoc validation error in NTOverlayItemTranslatorNodeTest
- b28b04 Fix javadoc validation error in InvOverlayItemTranslatorNodeTest
- 6ffa4b Fix javadoc validation error in I2RSOverlayItemTranslatorNodeTest
- da9651 Fix javadoc validation error in PathTranslatorTest
- 17a9e0 Fix javadoc validation error in UnificationCustomScriptTest
- decf1f Fix javadoc validation error in UnificationAggregatorTest
- fc3e81 Fix javadoc validation error in EqualityAggregatorTest
- d3d1a9 Bugfix for BUG-6095
- 8b4b5b Little performance improvement in calculation of updated links
- 4b7cee Bugfix for bug where link aggregation was not processed
- d95d66 Fix for BUG-6004
- 1d74d4 Bugfix for BUG-6055
- fe0998 BUG-5955
- 4a9637 Bugfix for bug5362

- 148b3c Fix for BUG-5862

Unified Secure Channel (USC)

- 0dfbbd Enable Maven Site generation
- af4cdb Improve remove channel and session
- 5f558b Improve UDP functions

User Network Interface Manager (UNIMGR)

- b880d2 merge the fix of the uni it test instability which was fixed by donald in the master to the stable/berilium branch.

VPN Service

- 67d44d Fix for upstream quagga: vtysh using #
- af2962 Added server mac address to Endpoint

Virtual Tenant Network (VTN)

- ec48c4 Fixed UT failure due to incorrect test data.
- 13b105 BUG-6258: Disabled VSEM Provider build.
- 1797d2 BUG-6166: Handle jeopardy state of clustered DS.
- 0a2481 BUG-6143: Fixed bug that failed to put vtn-data-flow into clustered DS.
- c045b7 Removed distributionManagement section from common
- adeb9a Disable VSEM Provider Build
- a3a273 BUG-5993: Ignore flow-removed that contains non-zero table ID.
- 8f02fc Fixed bug in mock-up for integration test.
- 06b427 BUG-5557: Fixed issue in delete operation for vbrif out flowfilter
- 3278a9 Fixed bug in mock-up for integration test.
- f27d82 Use https to connect ODL Nexus

YANG Tools

- a666f3 BUG-6195: Fix issue with leafSetEntryNode in SchemaTracker
- 922054 BUG-6173: Allow refine statement to have no substatements
- b19410 Add a couple of toString() implementations
- e0a42d BUG-5830: Mandatory leaf enforcement is not correct with presence container
- 73ecac Speed up toString() for XML elements
- 635af1 BUG-5200: Yang parser doesn't fill error-app-tag and error-message in constraints
- 594ac3 BUG-5899: Cardinality check incorrectly limited to 1 for "must"

- 0d9b03 BUG-5693: IOException after first parsing phase - stream closed
- 98ac2b BUG-5819 - Violation of synchronization rules in AbstractSchemaRepository
- 6d3f5a Do not advertize features-test in artifacts
- 87edd5 BUG-5446: Yangtools UnionStringCodec is not consistent with BinaryStringCodec
- 2ade6e BUG-5484: Fix of Yang statement lexer

Beryllium-SR4 Release Notes

This page details changes and bug fixes between the Beryllium Stability Release 3 (Beryllium-SR3) and the Beryllium Stability Release 4 (Beryllium-SR4) of OpenDaylight.

Projects with No Noteworthy Changes

The following projects had no noteworthy changes in the Beryllium-SR4 Release:

- ALTO
- Centinel
- Control And Provisioning of Wireless Access Points (CAPWAP)
- Controller Shield
- DLUX
- Device Identification and Driver Management (DIDM)
- Fabric As A Service (FaaS)
- Group Based Policy (GBP)
- Internet of Things Data Management (IoTDM)
- Link Aggregation Control Protocol (LACP)
- Messaging4Transport
- NEtwork MOdeling (NEMO)
- NeXt UI Toolkit
- NetIDE
- Network Intent Composition (NIC)
- Neutron Northbound
- Packet Cable/PCMM
- SDN Interface Application (SDNi)
- SNMP Plugin
- SNMP4SDN
- Secure Network Bootstrapping Infrastructure (SNBI)
- Secure tag eXchange Protocol (SXP)
- Service Function Chaining
- TCP-MD5

- Time Series Data Repository (TSDR)
- User Network Interface Manager (UNIMGR)
- VPN Service
- YANG PUBSUB

Authentication, Authorization and Accounting (AAA)

- a7853f Auto-detect secure HTTP in the idmtool script
- cbb7e9 Add support for Active Directory to AAA
- 9c9c51 Add groupRolesMap configuration option for ODLJndiLdapRealm
- f6a970 Do not override odlparent's version declaration

BGP PCEP

- 0cc5a7 BUG-6810: Fix intermintent Be BmpMonitorImplTest failure
- a67ad3 BUG-6739 - Throw exception while Installing odl-bgpcep-bgp-all feature
- cb5d14 BUG-6585: BGP ChannelOutputLimiter waits forever
- e65e13 BUG-6662: On connection reset by peer, sometimes re-connection attempt stops after HoldTimer expired error
- a47f8b BUG-6568 - Termination Point Type is never set in BGP-LS topology
- b549f6 BUG-6507: MalformedObjectNameException while configuring OpenConfig BGP IPv6 neighbor - Since colon is not a valid character for MBean key, replace colon in IPv6 neighbor address with dash while creating BGP Peer module name - Convert neighbor IPv6 address and key which is derieved from it to to normalized notation (full-form and leading 0s removed) to match with similar conversion for BGP Peer module done in BGPPeerModule#getNormalizedHost
- 2074c9 BUG-5922: inbound IPv6 connection fails if address has zero groups in it - Convert the remote IPv6 address to normalized notation (full-form and leading 0s removed) so that comparision with configured BGP Peer address works - BGP Peer address is converted to this form in BGPPeerModule#getNormalizedHost
- bf4a8c BUG-6242: PCRpt received with bandwidth reoptimization object leads to loop causing OOM - added bandwidth reoptimization object to the list of possible objects in PCRpt message - gracefully handle condition where unexpected object is present in received message - updated unit-tests
- badc28 BUG-6330: Fix race condition when reinstalling App Peer
- 9b3951 BUG-6342: Link-state topology takes long time to get updated - When large number of prefixes are present for node, repeated re-writing of node causes slowness - Avoid writing the entire node again when prefix or termination-point information for that node changes - Just write the modified information in such cases

Controller

- 31364d BUG-6348 : car:stop-stress-test RPC to return success & failure counters
- c17f2e Fix missing LeaderStateChanged event

Integration/Distribution

- 67eb4f Added a variation of the ‘configure-cluster.sh’ script that does not require the ‘index’ parameter. The script determines the local machine’s IP address and checks whether it’s present in ‘seed_nodes_list’. The script also determines the index based on the local IP address.
- e30ccb Set 2nd version for autorelease scripting
- 04d0d9 Add script to enable/disable config datastore persistence

L2 Switch

- e8a730 BUG-6655 - arphandler unable to flood arp packet
- ae78b3 BUG-6751 - l2switch project build failed in stable/beryllium

LISP Flow Mapping

- 093ff7 BUG-6634: Neutron - PortHandler throws NPE / Berryllium

MD-SAL

- 18138b BUG-2332: BindingMapping to camel split also on forward slash
- c91e99 BUG-6184: Workaround for namespaces with URL with trailing slash
- c3ce38 BUG-6126: Use importedName for java.lang types in ClassTemplate

NETCONF

- ce1ff7 Do a proper disconnect when deleting a connector.
- c1b194 BUG-6272 - support RESTCONF PATCH for mounted NETCONF nodes
- 5eccda BUG-6256 - OpenDaylight RESTCONF XML selects wrong YANG model for southbound NETCONF
- 75c307 BUG-6797 - Fix deadlock on cached schema-changed notifications
- 2278a4 Add unit tests for sal-netconf-connector
- 40cfb8 BUG-6198 - Use sal-netconf-connector to connet device costs too much time

Network Virtualization

- 8b4b26 BUG-6066 - Updated the logging.

ODL Root Parent

- 90cddb BUG-6187: Force more maven resolution in karaf-plugin

OVSDB Integration

- c8a3d9 BUG-6692: use non-deprecated firstKeyOf() variant
- cb5076 BUG-6692: clean up MonitorRequestBuilder
- 192e4f Clean up SouthboundConstants
- 5af9c8 Fix clear bug related to "num" in JSON Node [Guava instead JDK Optional]
- 9c60c1 Fixed inappropriate WARN message.
- f7c33a BUG-6475: Removed unnecessary read operations.
- 4a6da8 BUG-6472 - JsonRpcEndpoint Reaper Thread's daemon property is not set
- 4767c3 BUG-6454: ProcUtils stdout should be grouped together
- d7fca5 BUG-6463 - Monitoring _version column on the ovsdb table is generating huge updates from switch
- b662df Filter out monitor updates for some columns
- a28379 BUG-6332 - Conflicting modification Exception for topology path
- 473f90 BUG-6336 using single tx for logical switch, macs
- 466e0a BUG-6352 - [SR3] RPC timeout in JsonRPCEndpoint is not configurable
- 130c4c added support for switch/port fault status for hwvtep schema changes.
- f2aa35 Docker-compose files for hwvtep IT, ovs 2.4/2.5
- 93474c BUG-5951 - Termination point config reconciliation

OpenFlow Configuration Protocol (OF-CONFIG)

- c70c3b Add the OF_CONFIG_OVS capabilities

OpenFlow Plugin

- 375e02 BUG-6118: making the OFentityListener aware of the InJeopardy() flag
- e308cf BUG-6513 Remove FD from registry immediately
- a606e2 Fix Ipv6 format compression
- 7f7514 BUG-6059: Moving Statistics Manager to DTCL
- 42292d BUG-6086 - Bulk-o-matic add flow RPC does not work
- 5d23e3 Device*Registry quickfix - add* failed
- 7ed2bb BUG-6058:Currently the Operational Datastore does not get cleaned up and the switch continues to persist if the node that is connected to the switch goes down. The patch addresses it.
- 940142 Move empty match constant into OFConstants

OpenFlow Protocol Library

- 5ef255 BUG-6646 Fix infinite reschedule of flush
- caca77 BUG-6638 Failed entries marked as completed also counted as completed
- 5e8b6a BUG-6744 - the parameters of the function of registerMeterBandSerializer need to be more refined

- f93dde BUG-6674 - the key of the serialization function registered by the vendor is not refinement enough

Table Type Patterns (TTP)

- 541801 Create non-root ttp-parent

Topology Processing Framework

- c02b04 Fix for BUG-6263
- 02894e BUG-6152 fix

Unified Secure Channel (USC)

- bcebca Do not override odlparent version of maven dependecy plugin

Virtual Tenant Network (VTN)

- 4e8bb1 BUG-6846: Fixed bug in VTN Coordinator shutdown sequence.
- 744b2e BUG-6632: Fixed VTN coordinator build error on Fedora 24.
- c565c5 Enable Maven Site for VTN

YANG Tools

- 07a461 BUG-4456: add RecursiveExtensionResolver
- de990f BUG-6757: revert fix for BUG-4456
- 354c04 BUG-6771: Problem with typedefs nested in augment
- 107826 BUG-6410: Fixed initialization of typedefs in rpc
- 531010 BUG-6180 - Parser: Backslash double-quote in double-quoted string not recognized
- 75c484 BUG-6497: Do not lose augmentation statement order
- 6d051f BUG-6497: Do not lose augmentation statement order
- 37962d BUG-6414: Fixed DataNodeIterator’s traverseModule method
- 0ecc2d BUG-5884: Augmenting a choice without a case results in no getter

Project-Specific Installation Guides

Centinel Installation Guide

This document is for the user to install the artifacts that are needed for using Centinel functionality in the OpenDaylight by enabling the default Centinel feature. Centinel is a distributed reliable framework for collection, aggregation and analysis of streaming data which is added in OpenDaylight Beryllium Release.

Overview

The Centinel project aims at providing a distributed, reliable framework for efficiently collecting, aggregating and sinking streaming data across Persistence DB and stream analyzers (e.g., Graylog, Elasticsearch, Spark, Hive). This framework enables SDN applications/services to receive events from multiple streaming sources (e.g., Syslog, Thrift, Avro, AMQP, Log4j, HTTP/REST).

In Beryllium, we develop a "Log Service" and plug-in for log analyzer (e.g., Graylog). The Log service process real time events coming from log analyzer. Additionally, we provide stream collector (Flume- and Sqoop-based) that collects logs from OpenDaylight and sinks them to persistence service (integrated with TSDR). Centinel also includes a RESTCONF interface to inject events to north bound applications for real-time analytic/network configuration. Further, a Centinel User Interface (web interface) will be available to operators to enable rules/alerts/dashboard etc.

Pre Requisites for Installing Centinel

- Recent Linux distribution - 64bit/16GB RAM
- Java Virtual Machine 1.7 or above
- Apache Maven 3.1.1 or above

Preparing for Installation

There are some additional pre-requisites for Centinel, which can be done by integrate Graylog server, Apache Drill, Apache Flume and HBase.

Graylog server2 Installation

- Install MongoDB
  - import the MongoDB public GPG key into apt:

    ```
    sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
    ```

  - Create the MongoDB source list:

    ```
    echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen
    ↪' | sudo tee /etc/apt/sources.list.d/mongodb.list
    ```

  - Update your apt package database:

    ```
    sudo apt-get update
    ```

  - Install the latest stable version of MongoDB with this command:

    ```
    sudo apt-get install mongodb-org
    ```

- Install Elasticsearch
  - Graylog2 v0.20.2 requires Elasticsearch v.0.90.10. Download and install it with these commands:

    ```
    cd ~; wget https://download.elasticsearch.org/elasticsearch/elasticsearch/
    ↪elasticsearch-0.90.10.deb
    sudo dpkg -i elasticsearch-0.90.10.deb
    ```

- We need to change the Elasticsearch cluster.name setting. Open the Elasticsearch configuration file:

```
sudo vi /etc/elasticsearch/elasticsearch.yml
```

- Find the section that specifies cluster.name. Uncomment it, and replace the default value with graylog2:

```
cluster.name: graylog2
```

- Find the line that specifies network.bind_host and uncomment it so it looks like this:

```
network.bind_host: localhost
script.disable_dynamic: true
```

- Save and quit. Next, restart Elasticsearch to put our changes into effect:

```
sudo service elasticsearch restart
```

- After a few seconds, run the following to test that Elasticsearch is running properly:

```
curl -XGET 'http://localhost:9200/_cluster/health?pretty=true'
```

* Install Graylog2 server
  - Download the Graylog2 archive to /opt with this command:

```
cd /opt; sudo wget https://github.com/Graylog2/graylog2-server/releases/
↪download/0.20.2/graylog2-server-0.20.2.tgz
```

  - Then extract the archive:

```
sudo tar xvf graylog2-server-0.20.2.tgz
```

  - Let's create a symbolic link to the newly created directory, to simplify the directory name:

```
sudo ln -s graylog2-server-0.20.2 graylog2-server
```

  - Copy the example configuration file to the proper location, in /etc:

```
sudo cp /opt/graylog2-server/graylog2.conf.example /etc/graylog2.conf
```

  - Install pwgen, which we will use to generate password secret keys:

```
sudo apt-get install pwgen
```

  - Now must configure the admin password and secret key. The password secret key is configured in graylog2.conf, by the password_secret parameter. Generate a random key and insert it into the Graylog2 configuration with the following two commands:

```
SECRET=$(pwgen -s 96 1)
sudo -E sed -i -e 's/password_secret =.*/password_secret = '$SECRET'/' /etc/
↪graylog2.conf

PASSWORD=$(echo -n password | shasum -a 256 | awk '{print $1}')
sudo -E sed -i -e 's/root_password_sha2 =.*/root_password_sha2 = '$PASSWORD'/
↪' /etc/graylog2.conf
```

  - Open the Graylog2 configuration to make a few changes: (sudo vi /etc/graylog2.conf):

```
rest_transport_uri = http://127.0.0.1:12900/
elasticsearch_shards = 1
```

- Now let's install the Graylog2 init script. Copy graylog2ctl to /etc/init.d:

```
sudo cp /opt/graylog2-server/bin/graylog2ctl /etc/init.d/graylog2
```

- Update the startup script to put the Graylog2 logs in /var/log and to look for the Graylog2 server JAR file in /opt/graylog2-server by running the two following sed commands:

```
sudo sed -i -e 's/GRAYLOG2_SERVER_JAR=\${GRAYLOG2_SERVER_JAR:=graylog2-server.
↪jar}/GRAYLOG2_SERVER_JAR=\${GRAYLOG2_SERVER_JAR:=\/opt\/graylog2-server\/
↪graylog2-server.jar}/' /etc/init.d/graylog2
sudo sed -i -e 's/LOG_FILE=\${LOG_FILE:=log\/graylog2-server.log}/LOG_FILE=\$
↪{LOG_FILE:=\/var\/log\/graylog2-server.log}/' /etc/init.d/graylog2
```

- Install the startup script:

```
sudo update-rc.d graylog2 defaults
```

- Start the Graylog2 server with the service command:

```
sudo service graylog2 start
```

Install Graylog Server using Virtual Machine

- Download the OVA image from link given below and save it to your disk locally: https://github.com/Graylog2/graylog2-images/tree/master/ova
- Run the OVA in many systems like VMware or VirtualBox.

HBase Installation

- Download hbase-0.98.15-hadoop2.tar.gz
- Unzip the tar file using below command:

```
tar -xvf hbase-0.98.15-hadoop2.tar.gz
```

- Create directory using below command:

```
sudo mkdir /usr/lib/hbase
```

- Move hbase-0.98.15-hadoop2 to hbase using below command:

```
mv hbase-0.98.15-hadoop2/usr/lib/hbase/hbase-0.98.15-hadoop2 hbase
```

- Configuring HBase with java
  - Open your hbase/conf/hbase-env.sh and set the path to the java installed in your system:

```
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_25
```

  - Set the HBASE_HOME path in bashrc file
    - Open bashrc file using this command:

```
gedit ~/.bashrc
```

- In bashrc file append the below 2 statements:

```
export HBASE_HOME=/usr/lib/hbase/hbase-0.98.15-hadoop2

export PATH=$PATH:$HBASE_HOME/bin
```

- To start HBase issue following commands:

```
HBASE_PATH$ bin/start-hbase.sh

HBASE_PATH$ bin/hbase shell
```

- Create centinel table in HBase with stream,alert,dashboard and stringdata as column families using below command:

```
create 'centinel','stream','alert','dashboard','stringdata'
```

- To stop HBase issue following command:

```
HBASE_PATH$ bin/stop-hbase.sh
```

Apache Flume Installation

- Download apache-flume-1.6.0.tar.gz
- Copy the downloaded file to the directory where you want to install Flume.
- Extract the contents of the apache-flume-1.6.0.tar.gz file using below command. Use sudo if necessary:

```
tar -xvzf apache-flume-1.6.0.tar.gz
```

- Starting flume
  - Navigate to the Flume installation directory.
  - Issue the following command to start flume-ng agent:

```
./flume-ng agent --conf conf --conf-file multiplecolumn.conf --name a1 -
↪Dflume.root.logger=INFO,console
```

Apache Drill Installation

- Download apache-drill-1.1.0.tar.gz
- Copy the downloaded file to the directory where you want to install Drill.
- Extract the contents of the apache-drill-1.1.0.tar.gz file using below command:

```
tar -xvzf apache-drill-1.1.0.tar.gz
```

- Starting Drill:
  - Navigate to the Drill installation directory.
  - Issue the following command to launch Drill in embedded mode:

```
bin/drill-embedded
```

- Access the Apache Drill UI on link: http://localhost:8047/
- Go to "Storage" tab and enable "HBase" storage plugin.

Deploying plugins

- Use the following command to download git repository of Centinel:

```
git clone https://git.opendaylight.org/gerrit/p/centinel
```

- Navigate to the installation directory and build the code using maven by running below command:

```
mvn clean install
```

- After building the maven project, a jar file named `centinel-SplittingSerializer-0.0.1-SNAPSHOT.jar` will be created in `centinel/plugins/centinel-SplittingSerializer/target` inside the workspace directory. Copy and rename this jar file to `centinel-SplittingSerializer.jar` (as mentioned in configuration file of flume) and save at location `apache-flume-1.6.0-bin/lib` inside flume directory.
- After successful build, copy the jar files present at below locations to `/opt/graylog/plugin` in graylog server(VM):

```
centinel/plugins/centinel-alertcallback/target/centinel-alertcallback-0.1.0-
↪SNAPSHOT.jar

centinel/plugins/centinel-output/target/centinel-output-0.1.0-SNAPSHOT.jar
```

- Restart the server after adding plugin using below command:

```
sudo graylog-ctl restart graylog-server
```

Configure rsyslog

Make changes to following file:

```
/etc/rsyslog.conf
```

- Uncomment `$InputTCPServerRun 1514`
- Add the following lines:

```
module(load="imfile" PollingInterval="10") #needs to be done just once
input(type="imfile"
File="<karaf.log>" #location of log file
StateFile="statefile1"
Tag="tag1")
*.* @@127.0.0.1:1514 # @@used for TCP
```

  - Use the following format and comment the previous one:

```
$ActionFileDefaultTemplate RSYSLOG_SyslogProtocol23Format
```

- Use the below command to send Centinel logs to a port:

  ```
  tail -f <location of log file>/karaf.log|logger
  ```

- Restart rsyslog service after making above changes in configuration file:

  ```
  sudo service rsyslog restart
  ```

Install the following feature

Finally, from the Karaf console install the Centinel feature with this command:

```
feature:install odl-centinel-all
```

Verifying your Installation

If the feature install was successful you should be able to see the following Centinel commands added:

```
centinel:list

centinel:purgeAll
```

Troubleshooting

Check the `../data/log/karaf.log` for any exception related to Centinel related features

Upgrading From a Previous Release

Beryllium being the first release supporting Centinel functionality, only fresh installation is possible.

Uninstalling Centinel

To uninstall the Centinel functionality, you need to do the following from Karaf console:

```
feature:uninstall centinel-all
```

Its recommended to restart the Karaf container after uninstallation of the Centinel functionality.

OpFlex agent-ovs Install Guide

Required Packages

You'll need to install the following packages and their dependencies:

- libuv
- openvswitch-gbp
- openvswitch-gbp-lib
- openvswitch-gbp-kmod

- libopflex
- libmodelgbp
- agent-ovs

Packages are available for Red Hat Enterprise Linux 7 and Ubuntu 14.04 LTS. Some of the examples below are specific to RHEL7 but you can run the equivalent commands for upstart instead of systemd.

Note that many of these steps may be performed automatically if you're deploying this along with a larger orchestration system.

Host Networking Configuration

You'll need to set up your VM host uplink interface. You should ensure that the MTU of the underlying network is sufficient to handle tunneled traffic. We will use an example of setting up *eth0* as your uplink interface with a vlan of 4093 used for the networking control infrastructure and tunnel data plane.

We just need to set the MTU and disable IPv4 and IPv6 autoconfiguration. The MTU needs to be large enough to allow both the VXLAN header and VLAN tags to pass through without fragmenting for best performance. We'll use 1600 bytes which should be sufficient assuming you are using a default 1500 byte MTU on your virtual machine traffic. If you already have any NetworkManager connections configured for your uplink interface find the connection name and proceed to the next step. Otherwise, create a connection with (be sure to update the variable UPLINK_IFACE as needed):

```
UPLINK_IFACE=eth0
nmcli c add type ethernet ifname $UPLINK_IFACE
```

Now, configure your interface as follows:

```
CONNECTION_NAME="ethernet-$UPLINK_IFACE"
nmcli connection mod "$CONNECTION_NAME" connection.autoconnect yes \
    ipv4.method link-local \
    ipv6.method ignore \
    802-3-ethernet.mtu 9000 \
    ipv4.routes '224.0.0.0/4 0.0.0.0 2000'
```

Then bring up the interface with:

```
nmcli connection up "$CONNECTION_NAME"
```

Next, create the infrastructure interface using the infrastructure VLAN (4093 by default). We'll need to create a vlan subinterface of your uplink interface, the configure DHCP on that interface. Run the following commands. Be sure to replace the variable values if needed. If you're not using NIC teaming, replace the variable team0 below:

```
UPLINK_IFACE=team0
INFRA_VLAN=4093
nmcli connection add type vlan ifname $UPLINK_IFACE.$INFRA_VLAN dev $UPLINK_IFACE id
↪$INFRA_VLAN
nmcli connection mod vlan-$UPLINK_IFACE.$INFRA_VLAN \
    ethernet.mtu 1600 ipv4.routes '224.0.0.0/4 0.0.0.0 1000'
sed "s/CLIENT_ID/01:$(ip link show $UPLINK_IFACE | awk '/ether/ {print $2}')/" \
    > /etc/dhcp/dhclient-$UPLINK_IFACE.$INFRA_VLAN.conf <<EOF
send dhcp-client-identifier CLIENT_ID;
request subnet-mask, domain-name, domain-name-servers, host-name;
EOF
```

Now bring up the new interface with:

```
nmcli connection up vlan-$UPLINK_IFACE.$INFRA_VLAN
```

If you were successful, you should be able to see an IP address when you run:

```
ip addr show dev $UPLINK_IFACE.$INFRA_VLAN
```

OVS Bridge Configuration

We'll need to configure an OVS bridge which will handle the traffic for any virtual machines or containers that are hosted on the VM host. First, enable the openvswitch service and start it:

```
# systemctl enable openvswitch
ln -s '/usr/lib/systemd/system/openvswitch.service' '/etc/systemd/system/multi-user.
↪target.wants/openvswitch.service'
# systemctl start openvswitch
# systemctl status openvswitch
openvswitch.service - Open vSwitch
   Loaded: loaded (/usr/lib/systemd/system/openvswitch.service; enabled)
   Active: active (exited) since Fri 2014-12-12 17:20:13 PST; 3s ago
  Process: 3053 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 3053 (code=exited, status=0/SUCCESS)
Dec 12 17:20:13 ovs-server.cisco.com systemd[1]: Started Open vSwitch.
```

Next, we can create an OVS bridge (you may wish to use a different bridge name):

```
# ovs-vsctl add-br br0
# ovs-vsctl show
34aa83d7-b918-4e49-bcec-1b521acd1962
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.3.90"
```

Next, we configure a tunnel interface on our new bridge as follows:

```
# ovs-vsctl add-port br0 br0_vxlan0 -- \
    set Interface br0_vxlan0 type=vxlan \
    options:remote_ip=flow options:key=flow options:dst_port=8472
# ovs-vsctl show
34aa83d7-b918-4e49-bcec-1b521acd1962
    Bridge "br0"
        Port "br0_vxlan0"
            Interface "br0_vxlan0"
                type: vxlan
                options: {dst_port="8472", key=flow, remote_ip=flow}
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.3.90"
```

Open vSwitch is now configured and ready.

Agent Configuration

Before enabling the agent, we'll need to edit its configuration file, which is located at "/etc/opflex-agent-ovs/opflex-agent-ovs.conf".

First, we'll configure the Opflex protocol parameters. If you're using an ACI fabric, you'll need the OpFlex domain from the ACI configuration, which is the name of the VMM domain you mapped to the interface for this hypervisor. Set the "domain" field to this value. Next, set the "name" field to a hostname or other unique identifier for the VM host. Finally, set the "peers" list to contain the fixed static anycast peer address of 10.0.0.30 and port 8009. Here is an example of a completed section (bold text shows areas you'll need to modify):

```
"opflex": {
    // The globally unique policy domain for this agent.
    "domain": "[CHANGE ME]",

    // The unique name in the policy domain for this agent.
    "name": "[CHANGE ME]",

    // a list of peers to connect to, by hostname and port.  One
    // peer, or an anycast pseudo-peer, is sufficient to bootstrap
    // the connection without needing an exhaustive list of all
    // peers.
    "peers": [
        {"hostname": "10.0.0.30", "port": 8009}
    ],

    "ssl": {
        // SSL mode.  Possible values:
        // disabled: communicate without encryption
        // encrypted: encrypt but do not verify peers
        // secure: encrypt and verify peer certificates
        "mode": "encrypted",

        // The path to a directory containing trusted certificate
        // authority public certificates, or a file containing a
        // specific CA certificate.
        "ca-store": "/etc/ssl/certs/"
    }
},
```

Next, configure the appropriate policy renderer for the ACI fabric. You'll want to use a stitched-mode renderer. You'll need to configure the bridge name and the uplink interface name. The remote anycast IP address will need to be obtained from the ACI configuration console, but unless the configuration is unusual, it will be 10.0.0.32:

```
// Renderers enforce policy obtained via OpFlex.
"renderers": {
    // Stitched-mode renderer for interoperating with a
    // hardware fabric such as ACI
    "stitched-mode": {
        "ovs-bridge-name": "br0",

        // Set encapsulation type.  Must set either vxlan or vlan.
        "encap": {
            // Encapsulate traffic with VXLAN.
            "vxlan" : {
                // The name of the tunnel interface in OVS
                "encap-iface": "br0_vxlan0",
                // The name of the interface whose IP should be used
                // as the source IP in encapsulated traffic.
                "uplink-iface": "eth0.4093",

                // The vlan tag, if any, used on the uplink interface.
                // Set to zero or omit if the uplink is untagged.
                "uplink-vlan": 4093,

                // The IP address used for the destination IP in
                // the encapsulated traffic.  This should be an
                // anycast IP address understood by the upstream
                // stitched-mode fabric.
                "remote-ip": "10.0.0.32"
            }
        },
        // Configure forwarding policy
        "forwarding": {
            // Configure the virtual distributed router
            "virtual-router": {
                // Enable virtual distributed router.  Set to true
                // to enable or false to disable.  Default true.
                "enabled": true,

                // Override MAC address for virtual router.
                // Default is "00:22:bd:f8:19:ff"
                "mac": "00:22:bd:f8:19:ff",

                // Configure IPv6-related settings for the virtual
                // router
                "ipv6" : {
                    // Send router advertisement messages in
                    // response to router solicitation requests as
                    // well as unsolicited advertisements.
                    "router-advertisement": true
                }
            },

            // Configure virtual distributed DHCP server
            "virtual-dhcp": {
                // Enable virtual distributed DHCP server.  Set to
                // true to enable or false to disable.  Default
                // true.
                "enabled": true,

                // Override MAC address for virtual dhcp server.
                // Default is "00:22:bd:f8:19:ff"
                "mac": "00:22:bd:f8:19:ff"
            }
        },

        // Location to store cached IDs for managing flow state
        "flowid-cache-dir": "DEFAULT_FLOWID_CACHE_DIR"
    }
}
```

Finally, enable the agent service:

```
# systemctl enable agent-ovs
ln -s '/usr/lib/systemd/system/agent-ovs.service' '/etc/systemd/system/multi-user.
↪target.wants/agent-ovs.service'
# systemctl start agent-ovs
# systemctl status agent-ovs
agent-ovs.service - Opflex OVS Agent
   Loaded: loaded (/usr/lib/systemd/system/agent-ovs.service; enabled)
   Active: active (running) since Mon 2014-12-15 10:03:42 PST; 5min ago
 Main PID: 6062 (agent_ovs)
   CGroup: /system.slice/agent-ovs.service
           -6062 /usr/bin/agent_ovs
```

The agent is now running and ready to enforce policy. You can add endpoints to the local VM hosts using the OpFlex Group-based policy plugin from OpenStack, or manually.

OVSDB OpenStack Installation Guide

Overview

This guide is geared towards installing OpenDaylight to use the OVSDB project to provide Neutron support for OpenStack.

Open vSwitch (OVS) is generally accepted as the unofficial standard for Virtual Switching in the Open hypervisor based solutions. For information on OVS, see Open vSwitch.

With OpenStack within the SDN context, controllers and applications interact using two channels: OpenFlow and OVSDB. OpenFlow addresses the forwarding-side of the OVS functionality. OVSDB, on the other hand, addresses the management-plane. A simple and concise overview of Open Virtual Switch Database (OVSDB) is available at: http://networkstatic.net/getting-started-ovsdb/

Preparing for Installation

Follow the instructions in *Installing OpenDaylight*.

Note: By default the ODL OVSDB L3 forwarding is disabled. Enable the functionality by editing the `ovsdb.l3.fwd.enabled` setting and setting it to `yes`:

```
vi etc/custom.properties

ovsdb.l3.fwd.enabled=yes
```

Installing OVSDB OpenStack

Install the required features with the following command:

```
feature:install odl-ovsdb-openstack
```

Recognize that for the feature to fully install requires installing other required features and may take 30–60 seconds as those other features are automatically installed. The Karaf prompt will return before the feature is fully installed.

Verifying your Installation

To verify that the installation was successful, use the following commands in Karaf and verify that the required features have been installed:

```
opendaylight-user@root>feature:list -i | grep ovsdb
odl-ovsdb-openstack               | 1.2.1-Beryllium  | x          | ovsdb-1.2.1-
↪Beryllium                      | OpenDaylight :: OVSDB :: OpenStack Network Virtual
odl-ovsdb-library                 | 1.2.1-Beryllium  | x          | odl-ovsdb-library-
↪1.2.1-Beryllium          | OpenDaylight :: library
odl-ovsdb-southbound-api          | 1.2.1-Beryllium  | x          | odl-ovsdb-
↪southbound-1.2.1-Beryllium     | OpenDaylight :: southbound :: api
odl-ovsdb-southbound-impl         | 1.2.1-Beryllium  | x          | odl-ovsdb-
↪southbound-1.2.1-Beryllium     | OpenDaylight :: southbound :: impl
odl-ovsdb-southbound-impl-rest    | 1.2.1-Beryllium  | x          | odl-ovsdb-
↪southbound-1.2.1-Beryllium     | OpenDaylight :: southbound :: impl :: REST
odl-ovsdb-southbound-impl-ui      | 1.2.1-Beryllium  | x          | odl-ovsdb-
↪southbound-1.2.1-Beryllium     | OpenDaylight :: southbound :: impl :: UI

opendaylight-user@root>feature:list -i | grep neutron
odl-neutron-service               | 0.6.0-Beryllium  | x          | odl-neutron-0.6.0-
↪Beryllium              | OpenDaylight :: Neutron :: API
odl-neutron-northbound-api        | 0.6.0-Beryllium  | x          | odl-neutron-0.6.0-
↪Beryllium              | OpenDaylight :: Neutron :: Northbound
odl-neutron-spi                   | 0.6.0-Beryllium  | x          | odl-neutron-0.6.0-
↪Beryllium              | OpenDaylight :: Neutron :: API
odl-neutron-transcriber           | 0.6.0-Beryllium  | x          | odl-neutron-0.6.0-
↪Beryllium              | OpenDaylight :: Neutron :: Implementation

opendaylight-user@root>feature:list -i | grep openflowplugin
odl-openflowplugin-southbound     | 0.2.0-Beryllium  | x          | openflowplugin-0.2.
↪0-Beryllium             | OpenDaylight :: Openflow Plugin :: SouthBound
odl-openflowplugin-nsf-services   | 0.2.0-Beryllium  | x          | openflowplugin-0.2.
↪0-Beryllium             | OpenDaylight :: OpenflowPlugin :: NSF :: Services
odl-openflowplugin-nsf-model      | 0.2.0-Beryllium  | x          | openflowplugin-0.2.
↪0-Beryllium             | OpenDaylight :: OpenflowPlugin :: NSF :: Model
odl-openflowplugin-nxm-extensions | 0.2.0-Beryllium  | x          | openflowplugin-
↪extension-0.2.0-Beryllium | OpenDaylight :: Openflow Plugin :: Nicira Extension
```

Use the following command in karaf to view the logs. Verify that there are no errors logs relating to `odl-ovsdb-openstack`:

```
log:display
```

Look for the following log to indicate that the `odl-ovsdb-openstack` feature has been fully installed:

```
Successfully pushed configuration snapshot netvirt-providers-impl-default-config.
↪xml(odl-ovsdb-openstack,odl-ovsdb-openstack)
```

Troubleshooting

Reference the following link to the OVSDB NetVirt project wiki. The link has very helpful information for understanding the OVSDB Network Virtualization project:

- a link to a tutorial describing the project, it's goals, features and architecture.

- a link to a VirtualBox OVA file containing an all-in-one setup that you can simply import and see the OpenDaylight and OpenStack integration in action.
- slides describing how to use the OVA, run the demo and how to debug.
- a link to a Youtube presentation covering the slides and demo.

https://wiki.opendaylight.org/view/OVSDB_Integration:Main#Getting_Started_with_OpenDaylight_OVSDB_Plugin_Network_Virtualization

Uninstalling OVSDB OpenStack

Uninstall the `odl-ovsdb-openstack` feature by using the following command:

```
feature:uninstall odl-ovsdb-openstack
```

The shut down OpenDaylight with the following command:

```
system:shutdown
```

Use the following command to clean and reset the working state before starting OpenDaylight again:

```
rm -rf data/* journal/* snapshots/*
```

TSDR Installation Guide

This document is for the user to install the artifacts that are needed for using Time Series Data Repository (TSDR) functionality in the ODL Controller by enabling either an HSQLDB, HBase, or Cassandra Data Store.

Overview

The Time Series Data Repository (TSDR) project in OpenDaylight (ODL) creates a framework for collecting, storing, querying, and maintaining time series data in the OpenDaylight SDN controller. Please refer to the User Guide for the detailed description of the functionality of the project and how to use the corresponding features provided in TSDR.

Pre Requisites for Installing TSDR

The software requirements for TSDR HBase Data Store are as follows:

- In the case when the user chooses HBase or Cassandra data store, besides the software that ODL requires, we also require HBase and Cassandra database running in single node deployment scenario.

No additional software is required for the HSQLDB Data Stores.

Preparing for Installation

- When using HBase data store, download HBase from the following website:

  http://archive.apache.org/dist/hbase/hbase-0.94.15/

- When using Cassandra data store, download Cassandra 2.1 from the following website:

  https://cassandra.apache.org/download/

- No additional steps are required to install the TSDR HSQL Data Store.

Installing TSDR Data Stores

Installing HSQLDB Data Store

Once OpenDaylight distribution is up, from karaf console install the HSQLDB data store using the following command:

```
feature:install odl-tsdr-hsqldb-all
```

This will install hsqldb related dependency features (and can take sometime) as well as openflow statistics collector before returning control to the console.

Installing HBase Data Store

Installing TSDR HBase Data Store contains two steps:

1. Installing HBase server, and
2. Installing TSDR HBase Data Store features from ODL Karaf console.

In Beryllium, we only support HBase single node running together on the same machine as OpenDaylight. Therefore, follow the steps to download and install HBase server onto the same machine as where OpenDaylight is running:

1. Create a folder in Linux operating system for the HBase server. For example, create an hbase directory under `/usr/lib`:

   ```
   mkdir /usr/lib/hbase
   ```

2. Unzip the downloaded HBase server tar file.

   Run the following command to unzip the installation package:

   ```
   tar xvf <hbase-installer-name>  /usr/lib/hbase
   ```

3. Make proper changes in hbase-site.xml

   (a) Under <hbase-install-directory>/conf/, there is a hbase-site.xml. Although it is not recommended, an experienced user with HBase can modify the data directory for hbase server to store the data.

   (b) Modify the value of the property with name "hbase.rootdir" in the file to reflect the desired file directory for storing hbase data.

   The following is an example of the file:

   ```
   <configuration>
     <property>
       <name>hbase.rootdir</name>
       <value>file:///usr/lib/hbase/data</value>
     </property>
     <property>
       <name>hbase.zookeeper.property.dataDir</name>
       <value>/usr/lib/hbase/zookeeper</value>
     </property>
   </configuration>
   ```

4. start hbase server:

   ```
   cd <hbase-installation-directory>
   ./start-hbase.sh
   ```

5. start hbase shell:

```
cd <hbase-insatllation-directory>
./hbase shell
```

6. start Karaf console
7. install hbase data store feature from Karaf console:

```
feature:install odl-tsdr-hbase
```

Installing Cassandra Data Store

Installing TSDR Cassandra Data Store contains two steps:

1. Installing Cassandra server, and
2. Installing TSDR Cassandra Data Store features from ODL Karaf console.

In Beryllium, we only support Cassadra single node running together on the same machine as OpenDaylight. Therefore, follow these steps to download and install Cassandra server onto the same machine as where OpenDaylight is running:

1. Install Cassandra (latest stable version) by downloading the zip file and untar the tar ball to cassandra/ directory on the testing machine:

```
mkdir cassandra
wget http://www.eu.apache.org/dist/cassandra/2.1.10/apache-cassandra-2.1.10-bin.
↪tar.gz[2.1.10 is current stable version, it can vary]
mv apache-cassandra-2.1.10-bin.tar.gz cassandra/
cd cassandra
tar -xvzf apache-cassandra-2.1.10-bin.tar.gz
```

2. Start Cassandra from cassandra directory by running:

```
./apache-cassandra-2.1.10/bin/cassandra
```

3. Start cassandra shell by running:

```
./apache-cassandra-2.1.10/bin/cqlsh
```

4. Start Karaf according to the instructions above.
5. Install Cassandra data store feature from Karaf console:

```
feature:install odl-tsdr-cassandra
```

Verifying your Installation

After the TSDR data store is installed, no matter whether it is HBase data store, Cassandra data store, or HSQLDB data store, the user can verify the installation with the following steps.

1. Verify if the following two tsdr commands are available from Karaf console:

```
tsdr:list
tsdr:purgeAll
```

2. Verify if openflow statisitcs data can be received successfully:
   (a) Run "feature:install odl-tsdr-openflow-statistics-collector" from Karaf.
   (b) Run mininet to connect to ODL controller. For example, use the following command to start a three node topology:

   ```
   mn --topo single,3  --controller 'remote,ip=172.17.252.210,port=6653' --
   ↪switch ovsk,protocols=OpenFlow13
   ```

   (c) From Karaf console, the user should be able to retrieve the statistics data of OpenFlow statistics data from the console:

   ```
   tsdr:list FLOWSTATS
   ```

Troubleshooting

Check the `../data/log/karaf.log` for any exception related to TSDR features.

Post Installation Configuration

Post Installation Configuration for HSQLDB Data Store

The feature installation takes care of automated configuration of the datasource by installing a file in <install folder>/etc named org.ops4j.datasource-metric.cfg. This contains the default location of <install folder>/tsdr where the HSQLDB datastore files are stored. If you want to change the default location of the datastore files to some other location update the last portion of the url property in the org.ops4j.datasource-metric.cfg and then restart the Karaf container.

Post Installation Configuration for HBase Data Store

Please refer to HBase Data Store User Guide.

Post Installation Configuration for Cassandra Data Store

There is no post configuration for TSDR Cassandra data store.

Upgrading From a Previous Release

The HBase data store was supported in the previous release as well as in this release. However, we do not support data store upgrade for HBase data store. The user needs to reinstall TSDR and start to collect data in TSDR HBase datastore after the installation.

HSQLDB and Cassandra are new data stores introduced in this release. Therefore, upgrading from previous release does not apply in these two data store scenarios.

Uninstalling TSDR Data Stores

To uninstall TSDR HSQLDB data store

To uninstall the TSDR functionality with the default store, you need to do the following from karaf console:

```
feature:uninstall odl-tsdr-hsqldb-all
feature:uninstall odl-tsdr-core
feature:uninstall odl-tsdr-hsqldb
feature:uninstall odl-tsdr-openflow-statistics-collector
```

It is recommended to restart the Karaf container after the uninstallation of the TSDR functionality with the default store.

To uninstall TSDR HBase Data Store

To uninstall the TSDR functionality with the HBase data store,

- Uninstall HBase data store related features from karaf console:

  ```
  feature:uninstall odl-tsdr-hbase
  feature:uninstall odl-tsdr-core
  ```

- stop hbase server:

  ```
  cd <hbase-installation-directory>
  ./stop-hbase.sh
  ```

- remove the file directory that contains the HBase server installation:

  ```
  rm -r <hbase-installation-directory>
  ```

It is recommended to restart the Karaf container after the uninstallation of the TSDR data store.

To uninstall TSDR Cassandra Data Store

To uninstall the TSDR functionality with the Cassandra store,

- uninstall cassandra data store related features following from karaf console:

  ```
  feature:uninstall odl-tsdr-cassandra
  feature:uninstall odl-tsdr-core
  ```

- stop cassandra database:

  ```
  ps auwx | grep cassandra
  sudo kill pid
  ```

- remove the cassandra installation files:

  ```
  rm <cassandra-installation-directory>
  ```

It is recommended to restart the Karaf container after uninstallation of the TSDR data store.

VTN Installation Guide

Overview

OpenDaylight Virtual Tenant Network (VTN) is an application that provides multi-tenant virtual network on an SDN controller.

Conventionally, huge investment in the network systems and operating expenses are needed because the network is configured as a silo for each department and system. Therefore various network appliances must be installed for each tenant and those boxes cannot be shared with others. It is a heavy work to design, implement and operate the entire complex network.

The uniqueness of VTN is a logical abstraction plane. This enables the complete separation of logical plane from physical plane. Users can design and deploy any desired network without knowing the physical network topology or bandwidth restrictions.

VTN allows the users to define the network with a look and feel of conventional L2/L3 network. Once the network is designed on VTN, it will automatically be mapped into underlying physical network, and then configured on the individual switch leverage SDN control protocol. The definition of logical plane makes it possible not only to hide the complexity of the underlying network but also to better manage network resources. It achieves reducing reconfiguration time of network services and minimizing network configuration errors. OpenDaylight Virtual Tenant Network (VTN) is an application that provides multi-tenant virtual network on an SDN controller. It provides API for creating a common virtual network irrespective of the physical network.

It is implemented as two major components

- *VTN Manager*
- *VTN Coordinator*

VTN Manager

An OpenDaylight Plugin that interacts with other modules to implement the components of the VTN model. It also provides a REST interface to configure VTN components in OpenDaylight. VTN Manager is implemented as one plugin to the OpenDaylight. This provides a REST interface to create/update/delete VTN components. The user command in VTN Coordinator is translated as REST API to VTN Manager by the OpenDaylight Driver component. In addition to the above mentioned role, it also provides an implementation to the OpenStack L2 Network Functions API.

VTN Coordinator

The VTN Coordinator is an external application that provides a REST interface for an user to use OpenDaylight VTN Virtualization. It interacts with VTN Manager plugin to implement the user configuration. It is also capable of multiple OpenDaylight orchestration. It realizes VTN provisioning in OpenDaylight instances. In the OpenDaylight architecture VTN Coordinator is part of the network application, orchestration and services layer. VTN Coordinator will use the REST interface exposed by the VTN Manger to realize the virtual network using OpenDaylight. It uses OpenDaylight APIs (REST) to construct the virtual network in OpenDaylight instances. It provides REST APIs for northbound VTN applications and supports virtual networks spanning across multiple OpenDaylight by coordinating across OpenDaylight.

Preparing for Installation

VTN Manager

Follow the instructions in *Installing OpenDaylight*.

VTN Coordinator

1. Arrange a physical/virtual server with any one of the supported 64-bit OS environment.
   - RHEL 7
   - CentOS 7
   - Fedora 20 / 21 / 22
2. Install these packages:

```
yum install perl-Digest-SHA uuid libxslt libcurl unixODBC json-c bzip2
rpm -ivh http://yum.postgresql.org/9.3/redhat/rhel-6-x86_64/pgdg-redhat93-9.3-1.
↪noarch.rpm
yum install postgresql93-libs postgresql93 postgresql93-server postgresql93-
↪contrib postgresql93-odbc
```

Installing VTN

VTN Manager

Install Feature:

```
feature:install odl-vtn-manager-neutron odl-vtn-manager-rest
```

Note: The above command will install all features of VTN Manager. You can install only REST or Neutron also.

VTN Coordinator

- Enter into the externalapps directory in the top directory of Beryllium:

```
cd distribution-karaf-0.4.0-Beryllium/externalapps
```

- Run the below command to extract VTN Coordinator from the tar.bz2 file in the externalapps directory:

```
tar -C/ -jxvf distribution.vtn-coordinator-6.2.0-Beryllium-bin.tar.bz2
```

This will install VTN Coordinator to /usr/local/vtn directory. The name of the tar.bz2 file name varies depending on the version. Please give the same tar.bz2 file name which is there in your directory.

- Configuring database for VTN Coordinator:

```
/usr/local/vtn/sbin/db_setup
```

- To start the Coordinator:

```
/usr/local/vtn/bin/vtn_start
```

Using VTN REST API:

Get the version of VTN REST API using the below command, and make sure the setup is working:

```
curl --user admin:adminpass -H 'content-type: application/json' -X GET http://<VTN_
↪COORDINATOR_IP_ADDRESS>:8083/vtn-webapi/api_version.json
```

The response should be like this, but version might differ:

```
{"api_version":{"version":"V1.2"}}
```

Verifying your Installation

VTN Manager

- In the karaf prompt, type the below command to ensure that vtn packages are installed:

  ```
  feature:list | grep vtn
  ```

- Run any VTN Manager REST API:

  ```
  curl --user "admin":"admin" -H "Accept: application/json" -H "Content-type:␣
  ↪application/json" -X GET http://localhost:8181/restconf/operational/vtn:vtns
  ```

VTN Coordinator

```
ps -ef | grep unc will list all the vtn apps
Run any REST API for VTN Coordinator version
```

Uninstalling VTN

VTN Manager

```
feature:uninstall odl-vtnmanager-all
```

VTN Coordinator

1. Stop VTN:

   ```
   /usr/local/vtn/bin/vtn_stop
   ```

2. Remove the `usr/local/vtn` folder

Common OpenDaylight Features

OpenDaylight User Interface (DLUX)

This section introduces you to the OpenDaylight User Experience (DLUX) application.

Getting Started with DLUX

DLUX provides a number of different Karaf features, which you can enable and disable separately. In Boron they are:

1. odl-dlux-core
2. odl-dlux-node
3. odl-dlux-yangui
4. odl-dlux-yangvisualizer

Logging In

To log in to DLUX, after installing the application:

1. Open a browser and enter the login URL http://\protect\T1\textdollaryour-karaf-ip:8181/index.html in your browser (Chrome is recommended).
2. Login to the application with your username and password credentials.

Note: OpenDaylight's default credentials are *admin* for both the username and password.

Working with DLUX

After you login to DLUX, if you enable only odl-dlux-core feature, you will see only topology application available in the left pane.

Note: To make sure topology displays all the details, enable the odl-l2switch-switch feature in Karaf.

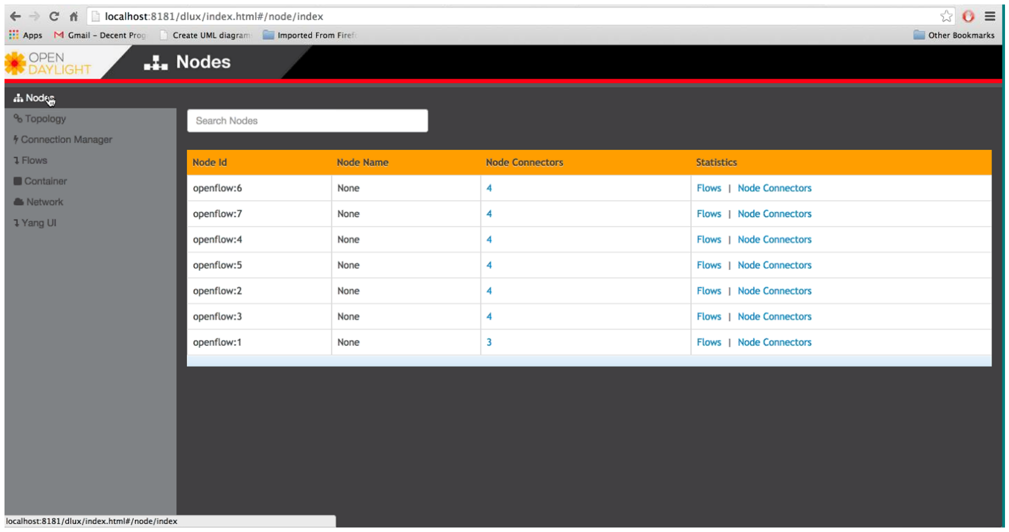
DLUX has other applications such as node, yang UI and those apps won't show up, until you enable their features odl-dlux-node and odl-dlux-yangui respectively in the Karaf distribution.

Note: If you install your application in dlux, they will also show up on the left hand navigation after browser page refresh.

Viewing Network Statistics

The *Nodes* module on the left pane enables you to view the network statistics and port information for the switches in the network.

To use the *Nodes* module:



Node Id	Node Name	Node Connectors	Statistics
openflow:6	None	4	Flows | Node Connectors
openflow:7	None	4	Flows | Node Connectors
openflow:4	None	4	Flows | Node Connectors
openflow:5	None	4	Flows | Node Connectors
openflow:2	None	4	Flows | Node Connectors
openflow:3	None	4	Flows | Node Connectors
openflow:1	None	3	Flows | Node Connectors

Fig. 1.1: DLUX Modules

1. Select *Nodes* on the left pane. The right pane displays atable that lists all the nodes, node connectors and the statistics.
2. Enter a node ID in the *Search Nodes* tab to search by node connectors.
3. Click on the *Node Connector* number to view details such as port ID, port name, number of ports per switch, MAC Address, and so on.
4. Click *Flows* in the Statistics column to view Flow Table Statistics for the particular node like table ID, packet match, active flows and so on.
5. Click *Node Connectors* to view Node Connector Statistics for the particular node ID.

Viewing Network Topology

The Topology tab displays a graphical representation of network topology created.

Note: DLUX does not allow for editing or adding topology information. The topology is generated and edited in other modules, e.g., the OpenFlow plugin. OpenDaylight stores this information in the MD-SAL datastore where DLUX can read and display it.

To view network topology:

1. Select *Topology* on the left pane. You will view the graphical representation on the right pane. In the diagram blue boxes represent the switches, the black represents the hosts available, and lines represents how the switches and hosts are connected.
2. Hover your mouse on hosts, links, or switches to view source and destination ports.
3. Zoom in and zoom out using mouse scroll to verify topology for larger topologies.

Interacting with the YANG-based MD-SAL datastore

The *Yang UI* module enables you to interact with the YANG-based MD-SAL datastore. For more information about YANG and how it interacts with the MD-SAL datastore, see the *Controller* and *YANG Tools* section of the *OpenDay-*

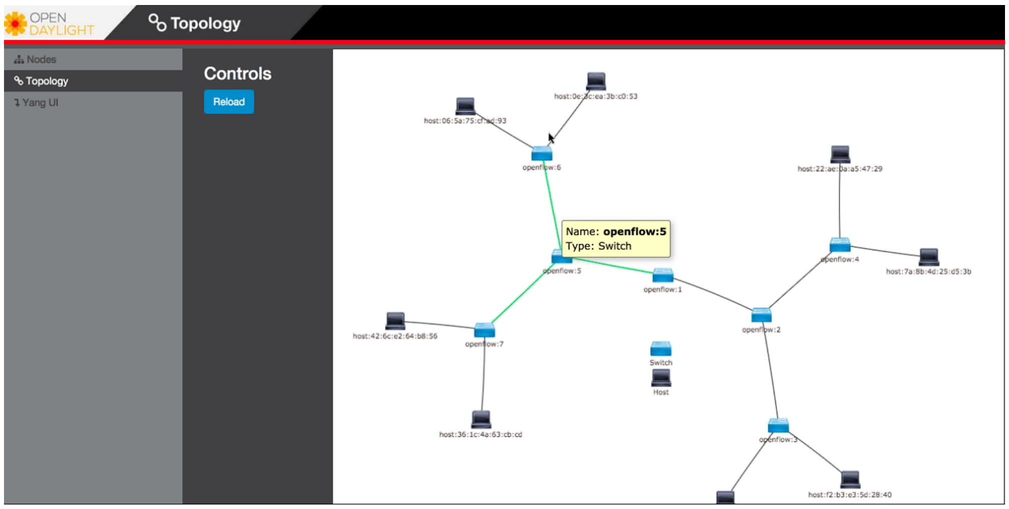
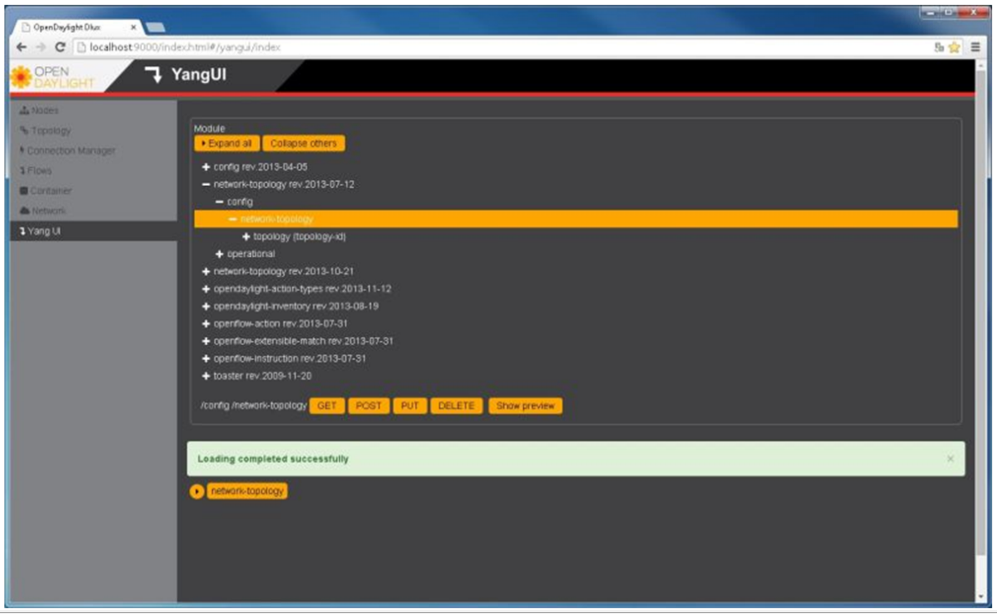


Fig. 1.2: Topology Module

light Developer Guide.



```
+ config rev.2013-04-05
− network-topology rev.2013-07-12
  − config
    − network-topology
      + topology {topology-id}
  + operational
+ network-topology rev.2013-10-21
+ opendaylight-action-types rev.2013-11-12
+ opendaylight-inventory rev.2013-08-19
+ openflow-action rev.2013-07-31
+ openflow-extensible-match rev.2013-07-31
+ openflow-instruction rev.2013-07-31
+ toaster rev.2009-11-20
/config /network-topology
```

Loading completed successfully

network-topology

Fig. 1.3: Yang UI

To use Yang UI:

1. Select *Yang UI* on the left pane. The right pane is divided in two parts.
2. The top part displays a tree of APIs, subAPIs, and buttons to call possible functions (GET, POST, PUT, and DELETE).

Note: Not every subAPI can call every function. For example, subAPIs in the *operational* store have GET functionality only.

Inputs can be filled from OpenDaylight when existing data from OpenDaylight is displayed or can be filled by user on the page and sent to OpenDaylight.

Buttons under the API tree are variable. It depends on subAPI specifications. Common buttons are:

- GET to get data from OpenDaylight,
- PUT and POST for sending data to OpenDaylight for saving
- DELETE for sending data to OpenDaylight for deleting.

You must specify the xpath for all these operations. This path is displayed in the same row before buttons and it may include text inputs for specific path element identifiers.

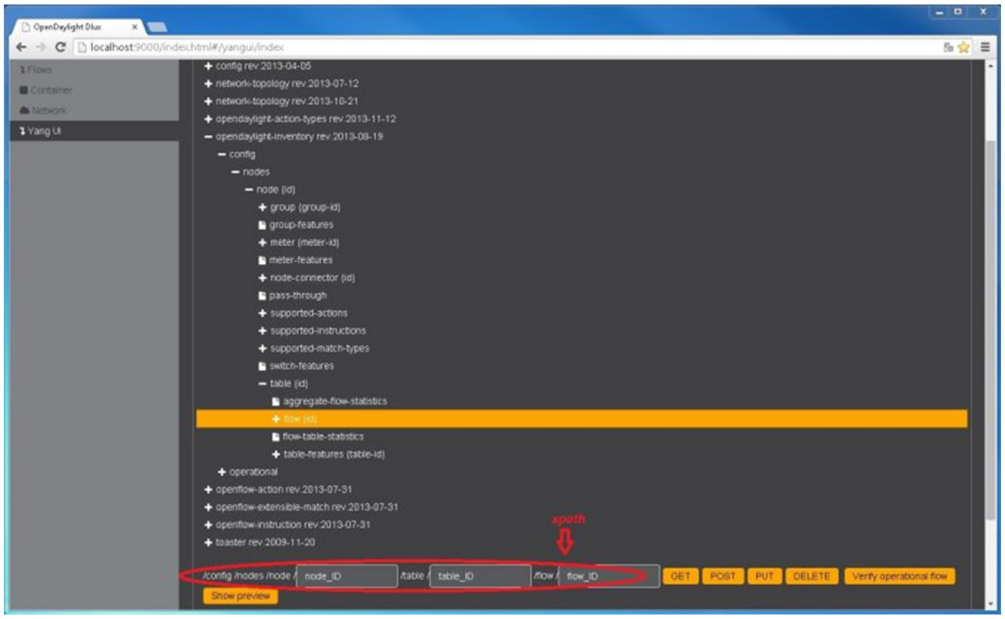

OpenDaylight Dlux
localhost:9000/index.html#/yangui/index
Flows
Container
Network
Yang UI
config rev.2013-04-05
network-topology rev.2013-07-12
network-topology rev.2013-10-21
opendaylight-action-types rev.2013-11-12
opendaylight-inventory rev.2013-08-19
config
nodes
node (id)
group (group-id)
group-features
meter (meter-id)
meter-features
node-connector (id)
pass-through
supported-actions
supported-instructions
supported-match-types
switch-features
table (id)
aggregate-flow-statistics
flow (id)
flow-table-statistics
table-features (table-id)
operational
openflow-action rev.2013-07-31
openflow-extensible-match rev.2013-07-31
openflow-instruction rev.2013-07-31
toaster rev.2009-11-20
xpath
/config /nodes /node / node_ID /table / table_ID /flow / flow_ID
GET
POST
PUT
DELETE
Verify operational flow
Show preview

Fig. 1.4: Yang API Specification

3. The bottom part of the right pane displays inputs according to the chosen subAPI.
   - Lists are handled as a special case. For example, a device can store multiple flows. In this case "flow" is name of the list and every list element is identified by a unique key value. Elements of a list can, in turn, contain other lists.
   - In Yang UI, each list element is rendered with the name of the list it belongs to, its key, its value, and a button for removing it from the list.
4. After filling in the relevant inputs, click the *Show Preview* button under the API tree to display request that will be sent to OpenDaylight. A pane is displayed on the right side with text of request when some input is filled.

Displaying Topology on the Yang UI

To display topology:

1. Select subAPI network-topology <topology revision number> == > operational == > network-topology.
2. Get data from OpenDaylight by clicking on the "GET" button.
3. Click *Display Topology*.

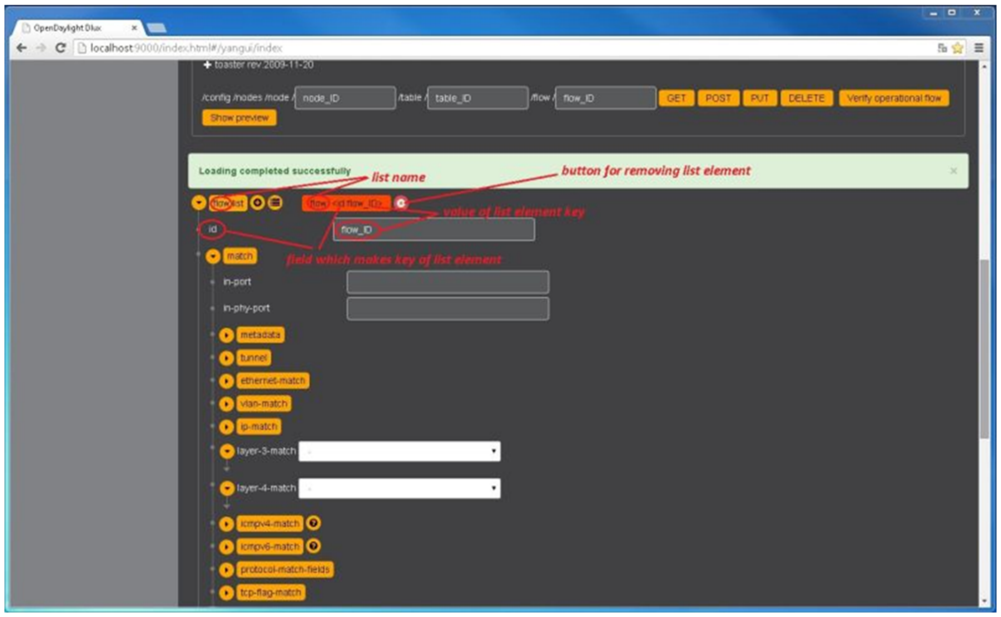


Fig. 1.5: Yang UI API Specification

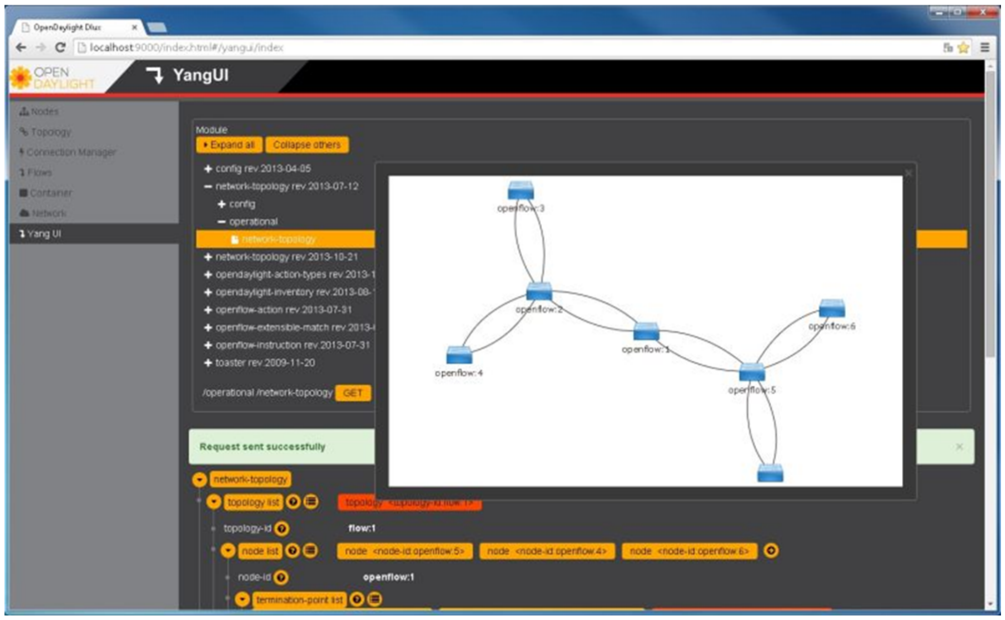


Fig. 1.6: DLUX Yang Topology

Configuring List Elements on the Yang UI

Lists in Yang UI are displayed as trees. To expand or collapse a list, click the arrow before name of the list. To configure list elements in Yang UI:

1. To add a new list element with empty inputs use the plus icon-button **+** that is provided after list name.
2. To remove several list elements, use the *X* button that is provided after every list element.

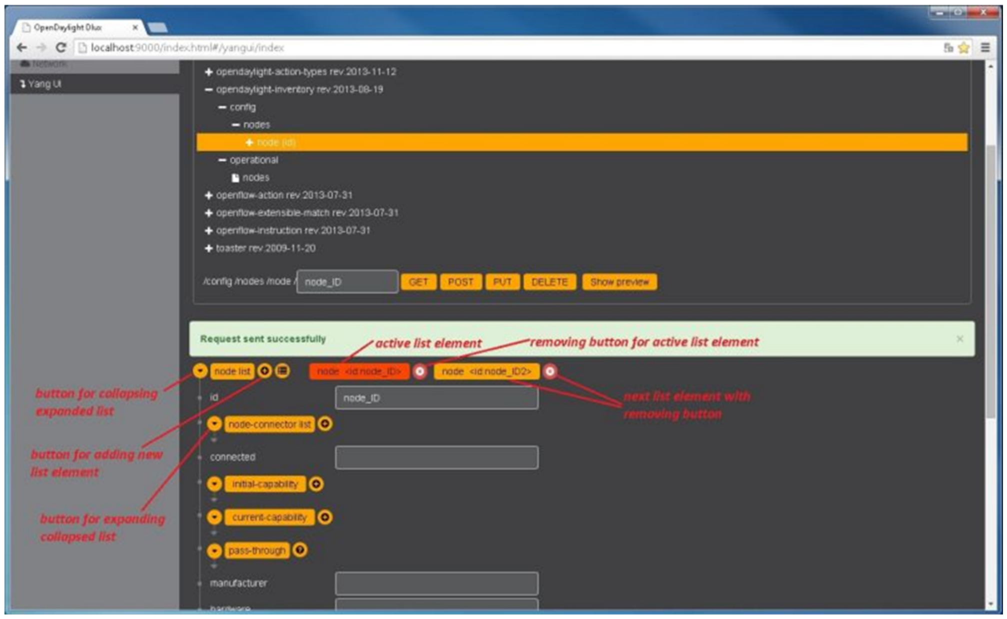

OpenDaylight Dlux
localhost:9000/index.html#/yangui/index
Network
Yang UI
opendaylight-action-types rev.2013-11-12
opendaylight-inventory rev.2013-08-19
config
nodes
node {id}
operational
nodes
openflow-action rev.2013-07-31
openflow-extensible-match rev.2013-07-31
openflow-instruction rev.2013-07-31
toaster rev.2009-11-20
/config /nodes /node /
node_ID
GET
POST
PUT
DELETE
Show preview
Request sent successfully
active list element
removing button for active list element
node list
node <id node_ID>
node <id node_ID2>
button for collapsing expanded list
id
node_ID
next list element with removing button
node-connector list
button for adding new list element
connected
initial-capability
current-capability
button for expanding collapsed list
pass-through
manufacturer
hardware

Fig. 1.7: DLUX List Elements

3. In the YANG-based data store all elements of a list must have a unique key. If you try to assign two or more elements the same key, a warning icon *!* is displayed near their name buttons.
4. When the list contains at least one list element, after the + icon, there are buttons to select each individual list element. You can choose one of them by clicking on it. In addition, to the right of the list name, there is a button which will display a vertically scrollable pane with all the list elements.

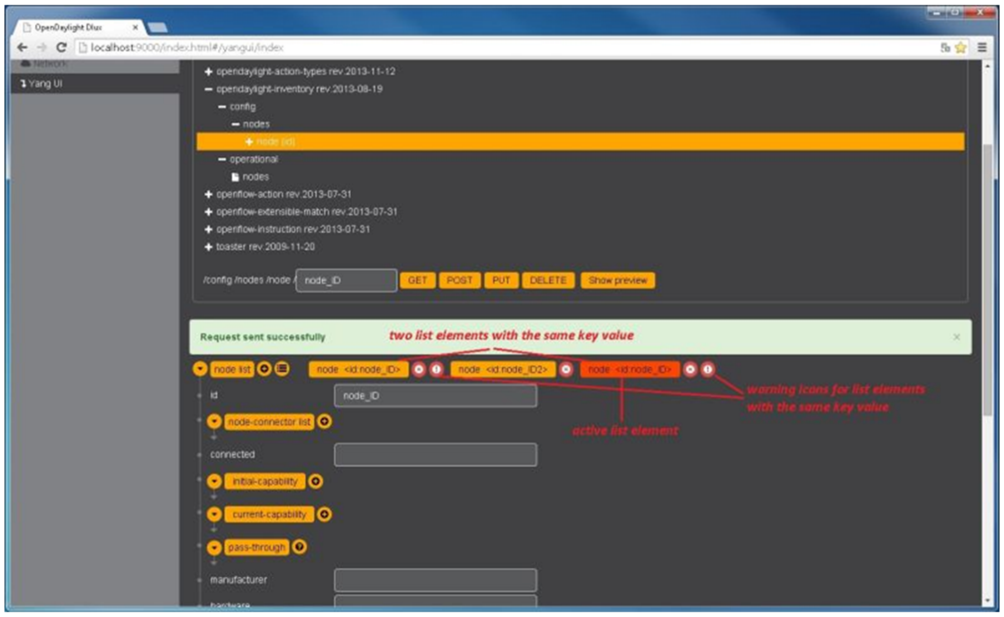
Setting Up Clustering

Clustering Overview

Clustering is a mechanism that enables multiple processes and programs to work together as one entity. For example, when you search for something on google.com, it may seem like your search request is processed by only one web server. In reality, your search request is processed by may web servers connected in a cluster. Similarly, you can have multiple instances of OpenDaylight working together as one entity.

Advantages of clustering are:

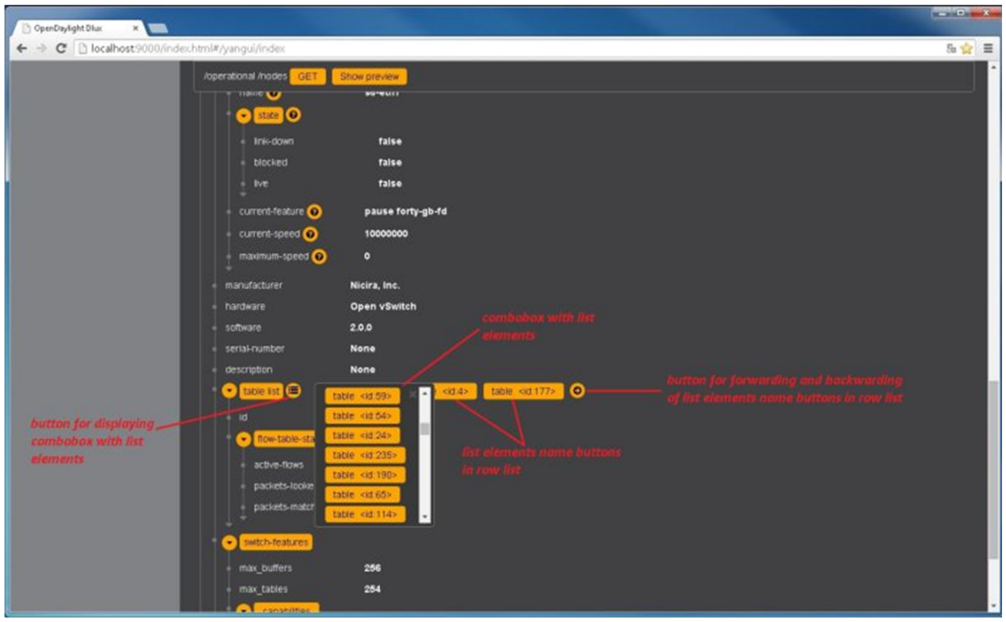
- Scaling: If you have multiple instances of OpenDaylight running, you can potentially do more work and store more data than you could with only one instance. You can also break up your data into smaller chunks (shards) and either distribute that data across the cluster or perform certain operations on certain members of the cluster.
- High Availability: If you have multiple instances of OpenDaylight running and one of them crashes, you will still have the other instances working and available.


localhost:9000/index.html#/yangui/index
Yang UI
opendaylight-action-types rev.2013-11-12
opendaylight-inventory rev.2013-08-19
config
nodes
node {id}
operational
nodes
openflow-action rev.2013-07-31
openflow-extensible-match rev.2013-07-31
openflow-instruction rev.2013-07-31
toaster rev.2009-11-20
/config /nodes /node /
GET
POST
PUT
DELETE
Show preview
two list elements with the same key value
node list
node <id:node_ID>
node <id:node_ID2>
node <id:node_ID>
warning icons for list elements with the same key value
active list element
node-connector list
initial-capability
current-capability
pass-through

Request sent successfully

id	node_ID
connected	
manufacturer	

Fig. 1.8: DLUX List Warnings


localhost:9000/index.html#/yangui/index
/operational /nodes
GET
Show preview
state
combobox with list elements
button for forwarding and backwarding of list elements name buttons in row list
button for displaying combobox with list elements
list elements name buttons in row list
table list
table <id:59>
table <id:54>
table <id:24>
table <id:235>
table <id:190>
table <id:65>
table <id:114>
<id:4>
table <id:177>
flow-table-sta
switch-features
capabilities

link-down	false
blocked	false
live	false
current-feature	pause forty-gb-fd
current-speed	10000000
maximum-speed	0
manufacturer	Nicira, Inc.
hardware	Open vSwitch
software	2.0.0
serial-number	None
description	None
id	
active-flows	
packets-looke	
packets-match	
max_buffers	256
max_tables	254

Fig. 1.9: DLUX List Button

- Data Persistence: You will not lose any data stored in OpenDaylight after a manual restart or a crash.

The following sections describe how to set up clustering on both individual and multiple OpenDaylight instances.

Multiple Node Clustering

The following sections describe how to set up multiple node clusters in OpenDaylight.

Deployment Considerations

To implement clustering, the deployment considerations are as follows:

- To set up a cluster with multiple nodes, we recommend that you use a minimum of three machines. You can set up a cluster with just two nodes. However, if one of the two nodes fail, the cluster will not be operational.

  Note: This is because clustering in OpenDaylight requires a majority of the nodes to be up and one node cannot be a majority of two nodes.

- Every device that belongs to a cluster needs to have an identifier. OpenDaylight uses the node's `role` for this purpose. After you define the first node's role as *member-1* in the `akka.conf` file, OpenDaylight uses *member-1* to identify that node.
- Data shards are used to contain all or a certain segment of a OpenDaylight's MD-SAL datastore. For example, one shard can contain all the inventory data while another shard contains all of the topology data.

  If you do not specify a module in the `modules.conf` file and do not specify a shard in `module-shards.conf`, then (by default) all the data is placed in the default shard (which must also be defined in `module-shards.conf` file). Each shard has replicas configured. You can specify the details of where the replicas reside in `module-shards.conf` file.
- If you have a three node cluster and would like to be able to tolerate any single node crashing, a replica of every defined data shard must be running on all three cluster nodes.

  Note: This is because OpenDaylight's clustering implementation requires a majority of the defined shard replicas to be running in order to function. If you define data shard replicas on two of the cluster nodes and one of those nodes goes down, the corresponding data shards will not function.

- If you have a three node cluster and have defined replicas for a data shard on each of those nodes, that shard will still function even if only two of the cluster nodes are running. Note that if one of those remaining two nodes goes down, the shard will not be operational.
- It is recommended that you have multiple seed nodes configured. After a cluster member is started, it sends a message to all of its seed nodes. The cluster member then sends a join command to the first seed node that responds. If none of its seed nodes reply, the cluster member repeats this process until it successfully establishes a connection or it is shut down.
- After a node is unreachable, it remains down for configurable period of time (10 seconds, by default). Once a node goes down, you need to restart it so that it can rejoin the cluster. Once a restarted node joins a cluster, it will synchronize with the lead node automatically.

Clustering Scripts

OpenDaylight includes some scripts to help with the clustering configuration.

Note: Scripts are stored in the OpenDaylight distribution/bin folder, and maintained in the distribution project repository in the folder distribution-karaf/src/main/assembly/bin/.

Configure Cluster Script

This script is used to configure the cluster parameters (e.g. akka.conf, module-shards.conf) on a member of the controller cluster. The user should restart the node to apply the changes.

Note: The script can be used at any time, even before the controller is started for the first time.

Usage:

```
bin/configure_cluster.sh <index> <seed_nodes_list>
```

- index: Integer within 1..N, where N is the number of seed nodes. This indicates which controller node (1..N) is configured by the script.
- seed_nodes_list: List of seed nodes (IP address), separated by comma or space.

The IP address at the provided index should belong to the member executing the script. When running this script on multiple seed nodes, keep the seed_node_list the same, and vary the index from 1 through N.

Optionally, shards can be configured in a more granular way by modifying the file "custom_shard_configs.txt" in the same folder as this tool. Please see that file for more details.

Example:

```
bin/configure_cluster.sh 2 192.168.0.1 192.168.0.2 192.168.0.3
```

The above command will configure the member 2 (IP address 192.168.0.2) of a cluster made of 192.168.0.1 192.168.0.2 192.168.0.3.

Setting Up a Multiple Node Cluster

To run OpenDaylight in a three node cluster, perform the following:

First, determine the three machines that will make up the cluster. After that, do the following on each machine:

1. Copy the OpenDaylight distribution zip file to the machine.
2. Unzip the distribution.
3. Open the following .conf files:
   - configuration/initial/akka.conf
   - configuration/initial/module-shards.conf
4. In each configuration file, make the following changes:

   Find every instance of the following lines and replace _127.0.0.1_ with the hostname or IP address of the machine on which this file resides and OpenDaylight will run:

   ```
   netty.tcp {
     hostname = "127.0.0.1"
   ```

Note: The value you need to specify will be different for each node in the cluster.

5. Find the following lines and replace _127.0.0.1_ with the hostname or IP address of any of the machines that will be part of the cluster:

```
cluster {
  seed-nodes = ["akka.tcp://opendaylight-cluster-data@127.0.0.1:2550",
                <url-to-cluster-member-2>,
                <url-to-cluster-member-3>]
```

6. Find the following section and specify the role for each member node. Here we assign the first node with the *member-1* role, the second node with the *member-2* role, and the third node with the *member-3* role:

```
roles = [
  "member-1"
]
```

Note: This step should use a different role on each node.

7. Open the configuration/initial/module-shards.conf file and update the replicas so that each shard is replicated to all three nodes:

```
replicas = [
    "member-1",
    "member-2",
    "member-3"
]
```

   For reference, view a sample config files <<_sample_config_files,below>>.

8. Move into the +<karaf-distribution-directory>/bin+ directory.
9. Run the following command:

```
JAVA_MAX_MEM=4G JAVA_MAX_PERM_MEM=512m ./karaf
```

10. Enable clustering by running the following command at the Karaf command line:

```
feature:install odl-mdsal-clustering
```

OpenDaylight should now be running in a three node cluster. You can use any of the three member nodes to access the data residing in the datastore.

Sample Config Files

Sample `akka.conf` file:

```
odl-cluster-data {
  bounded-mailbox {
    mailbox-type = "org.opendaylight.controller.cluster.common.actor.
↪MeteredBoundedMailbox"
    mailbox-capacity = 1000
    mailbox-push-timeout-time = 100ms
  }
  metric-capture-enabled = true

  akka {
    loglevel = "DEBUG"
    loggers = ["akka.event.slf4j.Slf4jLogger"]

    actor {

      provider = "akka.cluster.ClusterActorRefProvider"
      serializers {
                java = "akka.serialization.JavaSerializer"
                proto = "akka.remote.serialization.ProtobufSerializer"
              }

              serialization-bindings {
                  "com.google.protobuf.Message" = proto

              }
    }
    remote {
      log-remote-lifecycle-events = off
      netty.tcp {
        hostname = "10.194.189.96"
        port = 2550
        maximum-frame-size = 419430400
        send-buffer-size = 52428800
        receive-buffer-size = 52428800
      }
    }

    cluster {
      seed-nodes = ["akka.tcp://opendaylight-cluster-data@10.194.189.96:2550",
                    "akka.tcp://opendaylight-cluster-data@10.194.189.98:2550",
                    "akka.tcp://opendaylight-cluster-data@10.194.189.101:2550"]

      auto-down-unreachable-after = 10s

      roles = [
        "member-1"
      ]

    }
  }
}

odl-cluster-rpc {
  bounded-mailbox {
    mailbox-type = "org.opendaylight.controller.cluster.common.actor.
↪MeteredBoundedMailbox"
    mailbox-capacity = 1000
    mailbox-push-timeout-time = 100ms
  }

  metric-capture-enabled = true

  akka {
    loglevel = "INFO"
    loggers = ["akka.event.slf4j.Slf4jLogger"]

    actor {
      provider = "akka.cluster.ClusterActorRefProvider"

    }
    remote {
      log-remote-lifecycle-events = off
      netty.tcp {
        hostname = "10.194.189.96"
        port = 2551
      }
    }

    cluster {
      seed-nodes = ["akka.tcp://opendaylight-cluster-rpc@10.194.189.96:2551"]

      auto-down-unreachable-after = 10s
    }
  }
}
```

Sample `module-shards.conf` file:

```
module-shards = [
    {
        name = "default"
        shards = [
            {
                name="default"
                replicas = [
                    "member-1",
                    "member-2",
                    "member-3"
                ]
            }
        ]
    },
    {
        name = "topology"
        shards = [
            {
                name="topology"
                replicas = [
                    "member-1",
                    "member-2",
                    "member-3"
                ]
            }
        ]
    },
    {
        name = "inventory"
        shards = [
            {
                name="inventory"
                replicas = [
                    "member-1",
                    "member-2",
                    "member-3"
                ]
            }
        ]
    },
    {
        name = "toaster"
        shards = [
            {
                name="toaster"
                replicas = [
                    "member-1",
                    "member-2",
                    "member-3"
                ]
            }
        ]
    }
]
```

Set Persistence Script

This script is used to enable or disable the config datastore persistence. The default state is enabled but there are cases where persistence may not be required or even desired. The user should restart the node to apply the changes.

Note: The script can be used at any time, even before the controller is started for the first time.

Usage:

```
bin/set_persistence.sh <on/off>
```

Example:

```
bin/set_persistence.sh off
```

The above command will disable the config datastore persistence.

Running XSQL Console Commands and Queries

XSQL Overview

XSQL is an XML-based query language that describes simple stored procedures which parse XML data, query or update database tables, and compose XML output. XSQL allows you to query tree models like a sequential database. For example, you could run a query that lists all of the ports configured on a particular module and their attributes.

The following sections cover the XSQL installation process, supported XSQL commands, and the way to structure queries.

Installing XSQL

To run commands from the XSQL console, you must first install XSQL on your system:

1. Navigate to the directory in which you unzipped OpenDaylight
2. Start Karaf:

   ```
   ./bin/karaf
   ```

3. Install XSQL:

   ```
   feature:install odl-mdsal-xsql
   ```

XSQL Console Commands

To enter a command in the XSQL console, structure the command as follows:

```
odl:xsql _<XSQL command>_
```

The following table describes the commands supported in this OpenDaylight release.

Supported XSQL Console Commands

Command	*Description*
r	Repeats the last command you executed.
list vtables	Lists the schema node containers that are currently installed. Whenever an OpenDaylight module is installed, its YANG model is placed in the schema context. At that point, the XSQL receives a notification, confirms that the module's YANG model resides in the schema context and then maps the model to XSQL by setting up the necessary vtables and vfields. This command is useful when you need to determine vtable information for a query.
list vfields *<vtable name>*	Lists the vfields present in a specific vtable. This command is useful when you need to determine vfields information for a query.
jdbc *<ip address>*	When the ODL server is behind a firewall, and the JDBC client cannot connect to the JDBC server, run this command to start the client as a server and establish a connection.
exit	Closes the console.
tocsv	Enables or disables the forwarding of query output as a .csv file.
filename *<filename>*	Specifies the .tocsv file to which the query data is exported. If you do not specify a value for this option when the toccsv option is enabled, the filename for the query data file is generated automatically.

XSQL Queries

You can run a query to extract information that meets the criteria you specify using the information provided by the *list vtables* and *list vfields* _<vtable name>_ commands. Any query you run should be structured as follows:

select _<vfields you want to search for, separated by a comma and a space>_ *from* _<vtables you want to search in, separated by a comma and a space>_ *where* _<criteria>_ '*_<*criteria operator*>_';*

For example, if you want to search the nodes/node ID field in the nodes/node-connector table and find every instance of the Hardware-Address object that contains _BA_ in its text string, enter the following query:

```
select nodes/node.ID from nodes/node-connector where Hardware-Address like '%BA%';
```

The following criteria operators are supported:

Supported XSQL Query Criteria Operators

Criteria Operators	Description
=	Lists results that equal the value you specify.
!=	Lists results that do not equal the value you specify.
like	Lists results that contain the substring you specify. For example, if you specify *like %BC%*, every string that contains that particular substring is displayed.
<	Lists results that are less than the value you specify.
>	Lists results that are more than the value you specify.
and	Lists results that match both values you specify.
or	Lists results that match either of the two values you specify.
>=	Lists results that are more than or equal to the value you specify.
<=	Lists results that are less than or equal to the value you specify.
is null	Lists results for which no value is assigned.
not null	Lists results for which any value is assigned.
skip	Use this operator to list matching results from a child node, even if its parent node does not meet the specified criteria. See the following example for more information.

Example: Skip Criteria Operator

If you are looking at the following structure and want to determine all of the ports that belong to a YY type module:

- Network Element 1
  - Module 1, Type XX
    - Module 1.1, Type YY
      - Port 1
      - Port 2
  - Module 2, Type YY
    - Port 1
    - Port 2

If you specify *Module.Type='YY'* in your query criteria, the ports associated with module 1.1 will not be returned since its parent module is type XX. Instead, enter *Module.Type='YY' or skip Module!='YY'*. This tells XSQL to disregard any parent module data that does not meet the type YY criteria and collect results for any matching child modules. In this example, you are instructing the query to skip module 1 and collect the relevant data from module 1.1.

OpenDaylight Version

Overview

This feature allows NETCONF/RESTCONF users to determine the version of OpenDaylight they are communicating with.

Install the Version Feature

Follow these steps to install the version feature:

1. Navigate to the directory in which you installed OpenDaylight
2. Start Karaf:

```
./bin/karaf
```

3. Install Version feature:

```
feature:install odl-distribution-version
```

Note: For RESTCONF access, it is recommended to install odl-restconf and odl-netconf-connector-ssh.

Version Feature Usage

Example of RESTCONF request using curl from bash:

```
$ curl -u 'admin:admin' localhost:8181/restconf/config/network-topology:network-
↪topology/topology/topology-netconf/node/controller-config/yang-ext:mount/
↪config:modules/module/odl-distribution-version:odl-version/odl-distribution-version
```

Example response (formatted):

```
{
 "module": [
  {
   "type": "odl-distribution-version:odl-version",
   "name": "odl-distribution-version",
   "odl-distribution-version:version": "0.5.0-SNAPSHOT"
  }
 ]
}
```

Security Considerations

This document discusses the various security issues that might affect OpenDaylight. The document also lists specific recommendations to mitigate security risks.

This document also contains information about the corrective steps you can take if you discover a security issue with OpenDaylight, and if necessary, contact the Security Response Team, which is tasked with identifying and resolving security threats.

Overview of OpenDaylight Security

There are many different kinds of security vulnerabilities that could affect an OpenDaylight deployment, but this guide focuses on those where (a) the servers, virtual machines or other devices running OpenDaylight have been properly physically (or virtually in the case of VMs) secured against untrusted individuals and (b) individuals who have access, either via remote logins or physically, will not attempt to attack or subvert the deployment intentionally or otherwise.

While those attack vectors are real, they are out of the scope of this document.

What remains in scope is attacks launched from a server, virtual machine, or device other than the one running OpenDaylight where the attack does not have valid credentials to access the OpenDaylight deployment.

The rest of this document gives specific recommendations for deploying OpenDaylight in a secure manner, but first we highlight some high-level security advantages of OpenDaylight.

- Separating the control and management planes from the data plane (both logically and, in many cases, physically) allows possible security threats to be forced into a smaller attack surface.
- Having centralized information and network control gives network administrators more visibility and control over the entire network, enabling them to make better decisions faster. At the same time, centralization of network control can be an advantage only if access to that control is secure.

Note: While both previous advantages improve security, they also make an OpenDaylight deployment an attractive target for attack making understanding these security considerations even more important.

- The ability to more rapidly evolve southbound protocols and how they are used provides more and faster mechanisms to enact appropriate security mitigations and remediations.
- OpenDaylight is built from OSGi bundles and the Karaf Java container. Both Karaf and OSGi provide some level of isolation with explicit code boundaries, package imports, package exports, and other security-related features.
- OpenDaylight has a history of rapidly addressing known vulnerabilities and a well-defined process for reporting and dealing with them.

OpenDaylight Security Resources

- If you have any security issues, you can send a mail to *security@lists.opendaylight.org*.
- For the list of current OpenDaylight security issues that are either being fixed or resolved, refer to https://wiki.opendaylight.org/view/Security_Advisories.
- To learn more about the OpenDaylight security issues policies and procedure, refer to https://wiki.opendaylight.org/view/Security:Main

Deployment Recommendations

We recommend that you follow the deployment guidelines in setting up OpenDaylight to minimize security threats.

- The default credentials should be changed before deploying OpenDaylight.
- OpenDaylight should be deployed in a private network that cannot be accessed from the internet.
- Separate the data network (that connects devices using the network) from the management network (that connects the network devices to OpenDaylight).

Note: Deploying OpenDaylight on a separate, private management network does not eliminate threats, but only mitigates them. By construction, some messages must flow from the data network to the management network, e.g., OpenFlow *packet_in* messages, and these create an attack surface even if it is a small one.

- Implement an authentication policy for devices that connect to both the data and management network. These are the devices which bridge, likely untrusted, traffic from the data network to the management network.

Securing OSGi bundles

OSGi is a Java-specific framework that improves the way that Java classes interact within a single JVM. It provides an enhanced version of the *java.lang.SecurityManager* (ConditionalPermissionAdmin) in terms of security.

Java provides a security framework that allows a security policy to grant permissions, such as reading a file or opening a network connection, to specific code. The code maybe classes from the jarfile loaded from a specific URL, or a class signed by a specific key. OSGi builds on the standard Java security model to add the following features:

- A set of OSGi-specific permission types, such as one that grants the right to register an OSGi service or get an OSGi service from the service registry.
- The ability to dynamically modify permissions at runtime. This includes the ability to specify permissions by using code rather than a text configuration file.
- A flexible predicate-based approach to determining which rules are applicable to which *ProtectionDomain*. This approach is much more powerful than the standard Java security policy which can only grant rights based on a jarfile URL or class signature. A few standard predicates are provided, including selecting rules based upon bundle symbolic-name.
- Support for bundle *local permissions* policies with optional further constraints such as *DENY* operations. Most of this functionality is accessed by using the *OSGi ConditionalPermissionAdmin* service which is part of the OSGi core and can be obtained from the OSGi service registry. The *ConditionalPermissionAdmin* API replaces the earlier *PermissionAdmin* API.

For more information, refer to http://www.osgi.org/Main/HomePage.

Securing the Karaf container

Apache Karaf is a OSGi-based runtime platform which provides a lightweight container for OpenDaylight and applications. Apache Karaf uses either Apache Felix Framework or Eclipse Equinox OSGi frameworks, and provide additional features on top of the framework.

Apache Karaf provides a security framework based on Java Authentication and Authorization Service (JAAS) in compliance with OSGi recommendations, while providing RBAC (Role-Based Access Control) mechanism for the console and Java Management Extensions (JMX).

The Apache Karaf security framework is used internally to control the access to the following components:

- OSGi services
- console commands
- JMX layer
- WebConsole

The remote management capabilities are present in Apache Karaf by default, however they can be disabled by using various configuration alterations. These configuration options may be applied to the OpenDaylight Karaf distribution.

Note: Refer to the following list of publications for more information on implementing security for the Karaf container.

- For role-based JMX administration, refer to https://karaf.apache.org/manual/latest/monitoring.
- For remote SSH access configuration, refer to https://karaf.apache.org/manual/latest/remote.
- For WebConsole access, refer to https://karaf.apache.org/manual/latest/webconsole.
- For Karaf security features, refer to https://karaf.apache.org/manual/latest/security.

Disabling the remote shutdown port

You can lock down your deployment post installation. Set `karaf.shutdown.port=-1` in `etc/custom.properties` or `etc/config.properties` to disable the remote shutdown port.

Securing Southbound Plugins

Many individual southbound plugins provide mechanisms to secure their communication with network devices. For example, the OpenFlow plugin supports TLS connections with bi-directional authentication and the NETCONF plugin supports connecting over SSH. Meanwhile, the Unified Secure Channel plugin provides a way to form secure, remote connections for supported devices.

When deploying OpenDaylight, you should carefully investigate the secure mechanisms to connect to devices using the relevant plugins.

Securing OpenDaylight using AAA

AAA stands for Authentication, Authorization, and Accounting. All three of can help improve the security posture of and OpenDaylight deployment. In this release, only authentication is fully supported, while authorization is an experimental feature and accounting remains a work in progress.

The vast majority of OpenDaylight's northbound APIs (and all RESTCONF APIs) are protected by AAA by default when installing the +odl-restconf+ feature. In the cases that APIs are *not* protected by AAA, this will be noted in the per-project release notes.

By default, OpenDaylight has only one user account with the username and password *admin*. This should be changed before deploying OpenDaylight.

Security Considerations for Clustering

While OpenDaylight clustering provides many benefits including high availability, scale-out performance, and data durability, it also opens a new attack surface in the form of the messages exchanged between the various instances of OpenDaylight in the cluster. In the current OpenDaylight release, these messages are neither encrypted nor authenticated meaning that anyone with access to the management network where OpenDaylight exchanges these clustering messages can forge and/or read the messages. This means that if clustering is enabled, it is even more important that the management network be kept secure from any untrusted entities.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search