## Contents

1. Getting Started Guide .................................................. 3
2. Getting Started Guide (Markdown) ................................... 23
3. Indices and tables ......................................................... 25
1.1 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.

1.1.1 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.
1.1.2 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

1.2 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.

1.3 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.

1.4 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 OpenDaylight interfaces, the microservices architecture allows you to select available services, protocols, and REST APIs.

### 1.5 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.
1.5.1 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.

1.5.2 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.

**1.5.3 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.

**1.5.4 Border Gateway Monitoring Protocol (BMP)**

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

**1.5.5 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.

**1.5.6 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.

**1.5.7 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.

**1.5.8 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
1.5.9 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.

1.5.10 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

1.5.11 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.

1.5.12 Link Aggregation Control Protocol (LACP)

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

1.5.13 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.

1.5.14 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.

1.5.15 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 data-
  store
• odl-restconf: RESTCONF Northbound for MD-SAL and applications

1.5.16 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 OpenDay-
light. 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.

1.5.17 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 OpenStackSecurity 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

1.5.18 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.

1.5.19 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.

1.5.20 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.

1.5.21 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.

1.5.22 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

1.5.23 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.

1.5.24 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.

1.5.25 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.

1.5.26 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.
1.5.27 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.

1.5.28 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.

1.5.29 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 (SDNVVPN) project of Open Platform for NFV project (OPNFV) (available in Brahmaputra release)
• Network Overlay solution necessary for a Datacenter/Cloud environment

1.5.30 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.

1.6 OpenDaylight Experimental Features

- **Messaging4Transport**
- **Network Intent Composition (NIC)**
- **UNI Manager Plug-in (Unimgr)**
- **YANG-PUBSUB**

1.6.1 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.

1.6.2 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

1.6.3 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.

1.6.4 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.

1.7 Other features

1.7.1 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.

1.7.2 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.

### 1.8 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

### 1.9 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

#### 1.9.1 Install OpenDaylight

#### 1.9.2 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 you can access on the OpenDaylight Wiki: https://wiki.opendaylight.org/view/Project_list

- 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

1.9. Installing OpenDaylight
- 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 Beryllium, but intentionally do not have release notes:

- The Documentation Project produced this and the other downloadable documentation.
- The Integration Group hosted the OpenDaylight-wide tests and main release distribution.
- Controller Core Functionality Tutorials provided a single test suite (dsbenchmark) that was used as part of integration testing
- Release Engineering used autorelease to build the Beryllium release artifacts, including the main release download.
### 1.9.3 Beryllium features

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Feature Description</th>
<th>Karaf feature name</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Enables authentication with support for federation using Apache Shiro</td>
<td>odl-aaa-shiro</td>
<td>all</td>
</tr>
<tr>
<td>BGP</td>
<td>Provides support for Border Gateway Protocol (including Link-State Distribution) as a source of L3 topology information</td>
<td>odl-bgpcep-bgp</td>
<td>all</td>
</tr>
<tr>
<td>BMP</td>
<td>Provides support for BGP Monitoring Protocol as a monitoring station</td>
<td>odl-bgpcep-bmp</td>
<td>all</td>
</tr>
<tr>
<td>DIDM</td>
<td>Device Identification and Driver Management</td>
<td>odl-didm-all</td>
<td>all</td>
</tr>
<tr>
<td>Centinel</td>
<td>Provides interfaces for streaming analytics</td>
<td>odl-centinell-all</td>
<td>all</td>
</tr>
<tr>
<td>DLUX</td>
<td>Provides an intuitive graphical user interface for OpenDaylight</td>
<td>odl-dlux-all</td>
<td>all</td>
</tr>
<tr>
<td>Fabric as a Service (FaaS)</td>
<td>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.</td>
<td>odl-faaS-all</td>
<td>self+all</td>
</tr>
<tr>
<td>Group Based Policy (GBP)</td>
<td>Enables Endpoint Registry and Policy Repository REST APIs and associated functionality for Group Based Policy with the default renderer for OpenFlow renderers</td>
<td>odl-groupbasedpolicy-ofoverlay</td>
<td>self+all</td>
</tr>
<tr>
<td>GBP User Interface</td>
<td>Enables a web-based user interface for Group Based Policy</td>
<td>odl-groupbasedpolicy-ui</td>
<td>all</td>
</tr>
<tr>
<td>GBP FaaS renderer</td>
<td>Enables the Fabric as a Service renderer for Group Based Policy</td>
<td>odl-groupbasedpolicy-faaas</td>
<td>self+all</td>
</tr>
<tr>
<td>GBP Neutron Support</td>
<td>Provides OpenStack Neutron support using Group Based Policy</td>
<td>odl-groupbasedpolicy-neutronmapper</td>
<td>all</td>
</tr>
<tr>
<td>L2 Switch</td>
<td>Provides L2 (Ethernet) forwarding across connected OpenFlow switches and support for host tracking</td>
<td>odl-l2switch-switch-ui</td>
<td>self+all</td>
</tr>
<tr>
<td>LACP</td>
<td>Enables support for the Link Aggregation Control Protocol</td>
<td>odl-lacp-ui</td>
<td>self+all</td>
</tr>
<tr>
<td>LISP Flow Mapping</td>
<td>Enables LISP control plane services including the mapping system services REST API and LISP protocol SB plugin</td>
<td>odl-lispflowmapping-msmr</td>
<td>all</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Feature Description</td>
<td>Karaf feature name</td>
<td>Compatibility</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>NEMO CLI</td>
<td>Provides intent mapping and implementation with CLI for legacy devices</td>
<td>odl-nemo-cli-renderer</td>
<td>all</td>
</tr>
<tr>
<td>NEMO OpenFlow</td>
<td>Provides intent mapping and implementation for OpenFlow devices</td>
<td>odl-nemo-openflow-renderer</td>
<td>self+all</td>
</tr>
<tr>
<td>NetIDE</td>
<td>Enables portability and cooperation inside a single network by using a client/server multi-controller architecture</td>
<td>odl-netide-rest</td>
<td>all</td>
</tr>
<tr>
<td>NETCONF over SSH</td>
<td>Provides support to manage NETCONF-enabled devices over SSH</td>
<td>odl-netconf-connector-ssh</td>
<td>all</td>
</tr>
<tr>
<td>OF-CONFIG</td>
<td>Enables remote configuration of OpenFlow datapaths</td>
<td>odl-of-config-rest</td>
<td>all</td>
</tr>
<tr>
<td>OVSDB OpenStack Neutron</td>
<td>OpenStack Network Virtualization using OpenDaylight's OVSDB support</td>
<td>odl-ovsdb-openstack</td>
<td>all</td>
</tr>
<tr>
<td>OVSDB Southbound</td>
<td>OVSDB MDSAL southbound plugin for the Open_vSwitch schema</td>
<td>odl-ovsdb-southbound-impl-ui</td>
<td>all</td>
</tr>
<tr>
<td>OVSDB HWVTLP Southbound</td>
<td>OVSDB MDSAL hwvtlp southbound plugin for the hw_vtep schema</td>
<td>odl-ovsdb-hwvtlpsouthbound-ui</td>
<td>all</td>
</tr>
<tr>
<td>OVSDB NetVirt SFC</td>
<td>OVSDB NetVirt support for SFC</td>
<td>odl-ovsdb-sfc-ui</td>
<td>all</td>
</tr>
<tr>
<td>OpenFlow Flow Programming</td>
<td>Enables discovery and control of OpenFlow switches and the topology between them</td>
<td>odl-openflowplugin-flow-services-ui</td>
<td>all</td>
</tr>
<tr>
<td>OpenFlow Table Type Patterns</td>
<td>Allows OpenFlow Table Type Patterns to be manually associated with network elements</td>
<td>odl-ttp-all</td>
<td>all</td>
</tr>
<tr>
<td>Packetcable PCMM</td>
<td>Enables flow-based dynamic QoS management of CMTS use in the DCOS/6 infrastructure and a policy server</td>
<td>odl-packetcable-policy-server</td>
<td>self+all</td>
</tr>
<tr>
<td>PCEP</td>
<td>Enables support for PCEP</td>
<td>odl-bgpceppcep</td>
<td>all</td>
</tr>
<tr>
<td>RESTCONF API Support</td>
<td>Enables REST API access to the MD-SAL including the data store</td>
<td>odl-restconf</td>
<td>all</td>
</tr>
<tr>
<td>SDNInterface</td>
<td>Provides support for interaction and sharing of state between (non-clustered) OpenDaylight instances</td>
<td>odl-sdninterfaceapp-all</td>
<td>all</td>
</tr>
<tr>
<td>SFC over L2</td>
<td>Supports implementing Service Function Chaining using Layer 2 forwarding</td>
<td>odl-sfc-ofl2</td>
<td>self+all</td>
</tr>
<tr>
<td>SFC over LISP</td>
<td>Supports implementing Service Function Chaining using LISP</td>
<td>odl-sfclisp</td>
<td>all</td>
</tr>
<tr>
<td>SFC over REST</td>
<td>Supports implementing Service Function Chaining using REST CRUD operations on network elements</td>
<td>odl-sfc-sb-rest</td>
<td>all</td>
</tr>
<tr>
<td>SFC over VXLAN</td>
<td>Supports implementing Service Function Chaining using VXLAN tunnels</td>
<td>odl-sfc-ovs</td>
<td>self+all</td>
</tr>
<tr>
<td>SNMP Plugin</td>
<td>Enables monitoring and control of network elements via SNMP</td>
<td>odl-snmp-plugin</td>
<td>all</td>
</tr>
<tr>
<td>SNMP4SDN</td>
<td>Enables OpenFlow-like control of network elements via SNMP</td>
<td>odl-snmp4sdn-all</td>
<td>all</td>
</tr>
<tr>
<td>SSSD Federated Authentication</td>
<td>Enables support for federated authentication using SSSD</td>
<td>odl-aaa-sssd-plugin</td>
<td>all</td>
</tr>
<tr>
<td>Secure tag eXchange Protocol (SXP)</td>
<td>Enables distribution of shared tags to network devices</td>
<td>odl-sxp-controller</td>
<td>all</td>
</tr>
<tr>
<td>Time Series Data Repository (TSR)</td>
<td>Enables support for storing and querying time series data with the default data collector for OpenFlow statistics the default data store for HSQLDB</td>
<td>odl-tsdr-hsqldb-all</td>
<td>all</td>
</tr>
<tr>
<td>TSDR Data Collectors</td>
<td>Enables support for various TSDR data sources (collectors) including OpenFlow statistics, NetFlow statistics, SNMP data, Syslog, and OpenDaylight (controller) metrics</td>
<td>odl-tsdr-openflow-statistics-collector, odl-tsdr-flow-statistics-collector, odl-tsdr-snmp-data-collector</td>
<td>all</td>
</tr>
</tbody>
</table>
1.9.4 Other Beryllium features

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Description</th>
<th>Feature Name</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpFlex</td>
<td>Provides OpFlex Agent for Open VSwitch to enforce network policy, such as GBP, for locally-attached virtual machines or containers</td>
<td>n/a</td>
<td>all</td>
</tr>
<tr>
<td>NeXt</td>
<td>Provides a developer toolkit for designing network-centric topology user interfaces</td>
<td>n/a</td>
<td>all</td>
</tr>
</tbody>
</table>

1.9.5 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>
<thead>
<tr>
<th>Feature Name</th>
<th>Description</th>
<th>Karaf feature name</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Enables configurable role-based authorization</td>
<td>odl-aaa-authz</td>
<td>all</td>
</tr>
<tr>
<td>ALTO</td>
<td>Enables support for Application-Layer Traffic Optimization</td>
<td>odl-alto-core</td>
<td>self+all</td>
</tr>
<tr>
<td>CAPWAP</td>
<td>Enables control of supported wireless APs</td>
<td>odl-capwap-ac-rest</td>
<td>all</td>
</tr>
<tr>
<td>Clustered Authentication</td>
<td>Enables the use of the MD-SAL clustered data store for the authentication database</td>
<td>odl-aaa-authn-midal-cluster</td>
<td>all</td>
</tr>
<tr>
<td>Controller Shield</td>
<td>Provides controller security information to northbound applications</td>
<td>odl-usecplugin</td>
<td>all</td>
</tr>
<tr>
<td>GPB IO Visor Renderer</td>
<td>Provides support for rendering Group Based Policy to IO Visor</td>
<td>odl-groupbasedpolicy-visor</td>
<td>all</td>
</tr>
<tr>
<td>Internet of Things Data Management</td>
<td>Enables support for the oneM2M specification</td>
<td>odl-iotdm-onem2m</td>
<td>all</td>
</tr>
<tr>
<td>LISP Flow Mapping OpenStack Network Virtualization</td>
<td>Experimental support for OpenStack Neutron virtualization</td>
<td>odl-lispsflowmapping-neutron</td>
<td>self+all</td>
</tr>
<tr>
<td>Feature Name</td>
<td>Description</td>
<td>Karaf feature name</td>
<td>Compatibility</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Messaging4Transport</td>
<td>Introduces an AMQP Northbound to MD-SAL</td>
<td>odl-messaging4transport</td>
<td></td>
</tr>
<tr>
<td>Network Intent Composition (NIC)</td>
<td>Provides abstraction layer for communicating network intents (including a distributed intent mapping service REST API) using either Hazelcast or the MD-SAL as the backing data store for intents</td>
<td>odl-nic-hazelcast or odl-nic-core-mdsal</td>
<td>all</td>
</tr>
<tr>
<td>NIC Console</td>
<td>Provides a Karaf CLI extension for intent CRUD operations and mapping service operations</td>
<td>odl-nic-console</td>
<td></td>
</tr>
<tr>
<td>NIC VTN renderer</td>
<td>Virtual Tenant Network renderer for Network Intent Composition</td>
<td>odl-nic-renderer-vtn</td>
<td>self+all</td>
</tr>
<tr>
<td>NIC GBP renderer</td>
<td>Group Based Policy renderer for Network Intent Composition</td>
<td>odl-nic-renderer-gbp</td>
<td>self+all</td>
</tr>
<tr>
<td>NIC OpenFlow renderer</td>
<td>OpenFlow renderer for Network Intent Composition</td>
<td>odl-nic-renderer-of</td>
<td>self+all</td>
</tr>
<tr>
<td>NIC NEMO renderer</td>
<td>Network MOdeling renderer for Network Intent Composition</td>
<td>odl-nic-renderer-nemo</td>
<td>self+all</td>
</tr>
<tr>
<td>OVSDB NetVirt UI</td>
<td>OVSDB DLUX UI</td>
<td>odl-ovsdb-ui</td>
<td>all</td>
</tr>
<tr>
<td>Secure Networking Bootstrap</td>
<td>Defines a SNBI domain and associated white lists of devices to be accommodated to the domain</td>
<td>odl-snb-all</td>
<td>self+all</td>
</tr>
<tr>
<td>UNI Manager</td>
<td>Initiates the development of data models and APIs to facilitate configuration and provisioning connectivity services for OpenDaylight applications and services</td>
<td>odl-unimgr</td>
<td>all</td>
</tr>
<tr>
<td>YANG PUBSUB</td>
<td>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</td>
<td>odl-yangpush-rest</td>
<td>all</td>
</tr>
</tbody>
</table>

### 1.9.6 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.

### 1.9.7 Install the DLUX interface

OpenDaylight’s DLUX web interface draws information from topology and host databases to display information about the topology of the network, flow statistics, and host locations.

To integrate with OpenDaylight you must enable the DLUX Karaf feature. Each feature can be enabled or disabled separately. Ensure that you have created a topology and enabled the MD-SAL feature in the Karaf distribution before you use DLUX for network management. For more information about enabling the Karaf features for DLUX, refer to [Enable_DLUX_Feature](#).

### 1.9.8 MD-SAL clustering

In the Beryllium release and newer, the odl-mdsal-broker installs MD-SAL clustering automatically.
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.

### 2.1 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.

### 2.2 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

### 2.3 Beryllium features

Features to implement networking functionality provide release notes you can access on the OpenDaylight Wiki:

https://wiki.opendaylight.org/view/Project_list

- Authentication, Authorization and Accounting (AAA)
- ALTO
- BGPCEP
- Controller
Indices and tables

- genindex
- modindex
- search