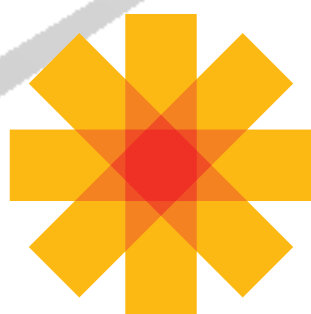




OPEN
DAYLIGHT

OpenDaylight Installation Guide

Beryllium (February 21, 2016)



OPEN
DAYLIGHT

OpenDaylight Installation Guide

OpenDaylight Community

Beryllium (2016-02-21)

Copyright © 2015 Linux Foundation All rights reserved.

This guide describes how to get started with OpenDaylight.

This program and the accompanying materials are made available under the terms of the Eclipse Public License v1.0 which accompanies this distribution, and is available at <http://www.eclipse.org/legal/epl-v10.html>

Table of Contents

1. Getting and Installing OpenDaylight	1
Downloading and installing OpenDaylight	1
Installing the components	2
Installing support for REST APIs	2
Installing MD-SAL clustering	2
I. Getting to know OpenDaylight	3
OpenDaylight Overview	5
2. Using the OpenDaylight User Interface (DLUX)	6
Getting Started with DLUX	6
Logging In	6
Working with DLUX	6
Viewing Network Statistics	7
Viewing Network Topology	7
Interacting with the YANG-based MD-SAL datastore	8
3. Running XSQL Console Commands and Queries	15
XSQL Overview	15
Installing XSQL	15
XSQL Console Commands	15
XSQL Queries	16
4. Setting Up Clustering	18
Clustering Overview	18
Single Node Clustering	18
Multiple Node Clustering	19
5. Security Considerations	24
Overview of OpenDaylight Security	24
OpenDaylight Security Resources	25
Deployment Recommendations	25
Securing OSGi bundles	26
Securing the Karaf container	26
Securing Southbound Plugins	27
Securing OpenDaylight using AAA	27
Security Considerations for Clustering	28
II. Project-specific Installation Guides	29
6. OpFlex agent-ovs Install Guide	31
Required Packages	31
Host Networking Configuration	31
OVS Bridge Configuration	32
Agent Configuration	33
7. OVSDB OpenStack Installation Guide	36
Overview	36
Preparing for Installation	36
Installing OVSDB OpenStack	36
Verifying your Installation	37
Uninstalling OVSDB OpenStack	37
8. OVSDB Service Function Chaining Installation Guide	38
Overview	38
Preparing for Installation	38
Installing OVSDB Service Function Chaining	38

Verifying your Installation	38
Uninstalling OVSDb Service Function Chaining	38
9. OVSDb NetVirt Hardware VTEP Installation Guide	40
Overview	40
Preparing for Installation	40
Installing OVSDb NetVirt Hardware VTEP	40
Verifying your Installation	40
Uninstalling OVSDb NetVirt Hardware VTEP	40
10. TSDR H2 Default Datastore Installation Guide	42
Overview	42
Pre Requisites for Installing TSDR with default H2 datastore	42
Preparing for Installation	42
Installing TSDR with default H2 datastore	42
Verifying your Installation	43
Post Installation Configuration	43
Upgrading From a Previous Release	43
Uninstalling TSDR with default H2 datastore	43
11. TSDR HBase Data Store Installation Guide	44
Overview	44
Prerequisites for Installing TSDR HBase Data Store	45
Preparing for Installation	45
Installing TSDR HBase Data Store	45
Verifying your Installation	46
Post Installation Configuration	46
Upgrading From a Previous Release	46
Uninstalling HBase Data Store	46
12. VTN Installation Guide	48
Overview	48
Preparing for Installation	49
Installing VTN	49
Verifying your Installation	50
Uninstalling VTN	51

List of Figures

2.1. DLUX Modules	7
2.2. Topology Module	8
2.3. Yang UI	9
2.4. Yang API Specification	10
2.5. Yang UI API Specification	11
2.6. DLUX Yang Topology	12
2.7. DLUX List Elements	13
2.8. DLUX List Warnings	13
2.9. DLUX List Button1	14

List of Tables

3.1. Supported XSQL Console Commands	15
3.2. Supported XSQL Query Criteria Operators	16

1. Getting and Installing OpenDaylight

Table of Contents

Downloading and installing OpenDaylight	1
Installing the components	2
Installing support for REST APIs	2
Installing MD-SAL clustering	2

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 [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
```





- 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.

Installing the components

The section describes a list of components in OpenDaylight Beryllium and the relevant Karaf feature to install in order to enable that component.

To install a feature use the following command:

```
feature:install
```

For Example:

```
feature:install <feature-name>
```

Multiple features can be installed using the following command:

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

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
```

Installing 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, you can activate this support by installing the `odl-restconf` feature.

Installing MD-SAL clustering

The MD-SAL clustering feature has "special" compatibility criteria. You **must** install clustering, before other features are installed. To install clustering, run the following command on the Karaf CLI console:

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


Part I. Getting to know OpenDaylight

Table of Contents

OpenDaylight Overview	5
2. Using the OpenDaylight User Interface (DLUX)	6
Getting Started with DLUX	6
Logging In	6
Working with DLUX	6
Viewing Network Statistics	7
Viewing Network Topology	7
Interacting with the YANG-based MD-SAL datastore	8
3. Running XSQL Console Commands and Queries	15
XSQL Overview	15
Installing XSQL	15
XSQL Console Commands	15
XSQL Queries	16
4. Setting Up Clustering	18
Clustering Overview	18
Single Node Clustering	18
Multiple Node Clustering	19
5. Security Considerations	24
Overview of OpenDaylight Security	24
OpenDaylight Security Resources	25
Deployment Recommendations	25
Securing OSGi bundles	26
Securing the Karaf container	26
Securing Southbound Plugins	27
Securing OpenDaylight using AAA	27
Security Considerations for Clustering	28

OpenDaylight Overview

The OpenDaylight project is a collaborative open source project that aims to accelerate adoption of Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) with a transparent approach that fosters new innovation.

OpenDaylight mainly consists of software designed to be run on top of a Java Virtual Machine (JVM) and can be run on any operating system and hardware as there is a Java Runtime Environment (JRE) available for it.

For a more detailed information about OpenDaylight, see the and *OpenDaylight User Guide*, *OpenDaylight Developer Guide*.

2. Using the OpenDaylight User Interface (DLUX)

Table of Contents

Getting Started with DLUX	6
Logging In	6
Working with DLUX	6
Viewing Network Statistics	7
Viewing Network Topology	7
Interacting with the YANG-based MD-SAL datastore	8

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 Beryllium they are: `. odl-dlux-core` `. odl-dlux-node` `. odl-dlux-yangui` `. odl-dlux-yangvisualizer`

Logging In

To log in to DLUX, after installing the application:

1. Open a browser and enter the login URL <http://<your-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

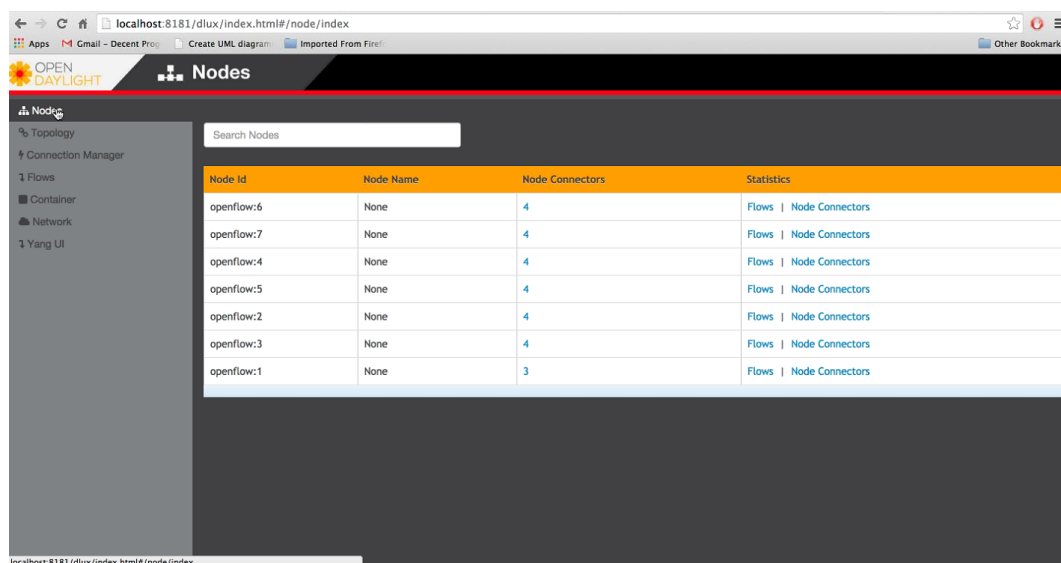
After you login to DLUX, if you enable only `odl-dlux-core` feature, you will see only topology application available in the left pane.



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.

Figure 2.1. DLUX Modules


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

**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:

1. Select **Nodes** on the left pane. The right pane displays a table 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.



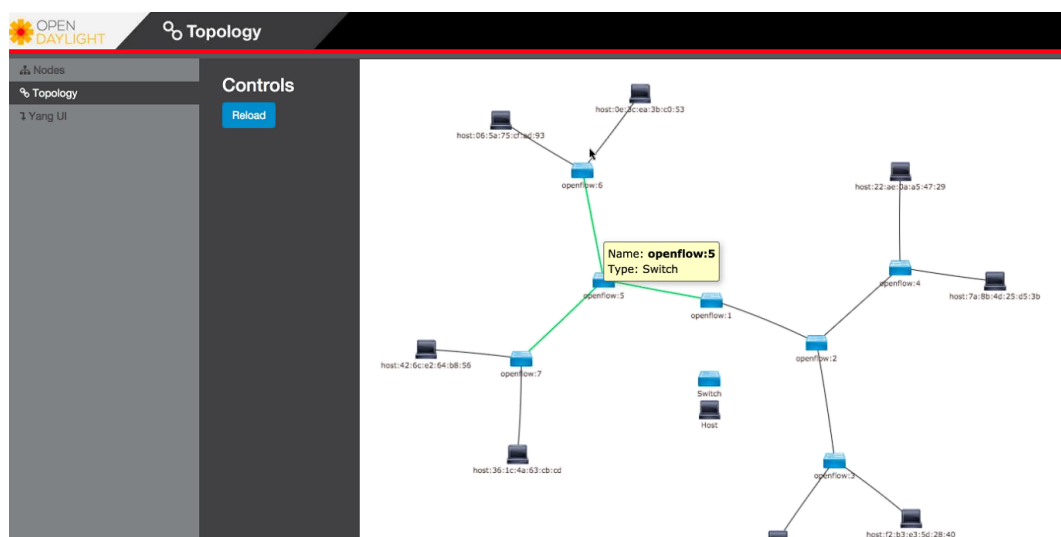
Note

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:

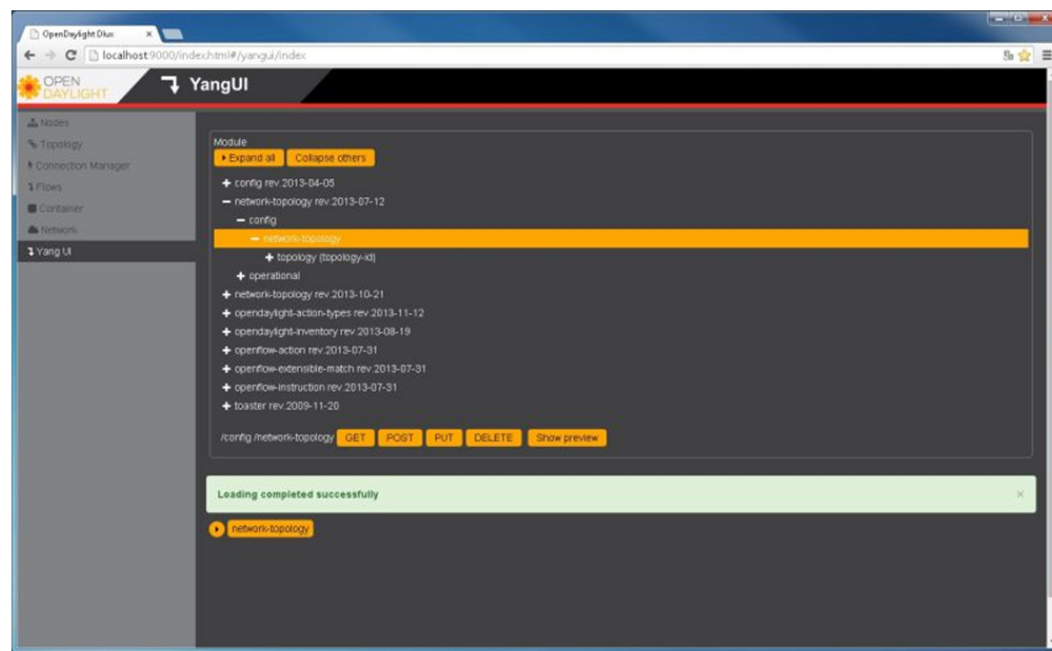
1. Select **Topology** on the left pane. You will view the graphical representation on the right pane. In the diagram blue boxes represent the switches, the black represents the hosts available, and lines represents how the switches and hosts are connected.
2. Hover your mouse on hosts, links, or switches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

Figure 2.2. Topology Module



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 *OpenDaylight Developer Guide*.

Figure 2.3. 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.

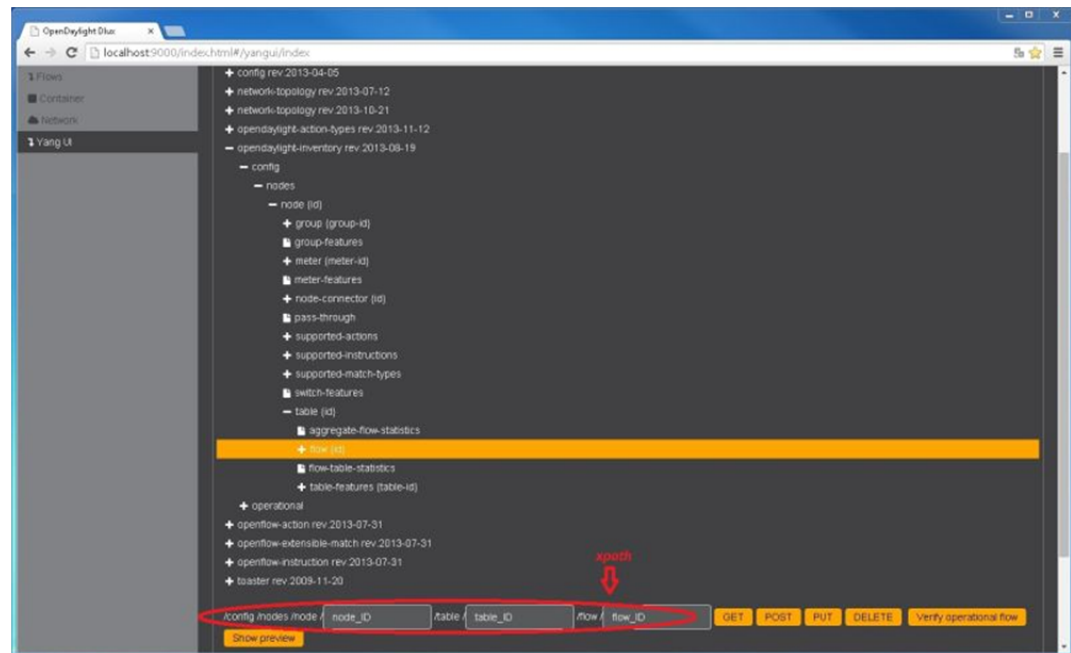
Inputs can be filled from OpenDaylight when existing data from OpenDaylight is displayed or can be filled by user on the page and sent to OpenDaylight.

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.

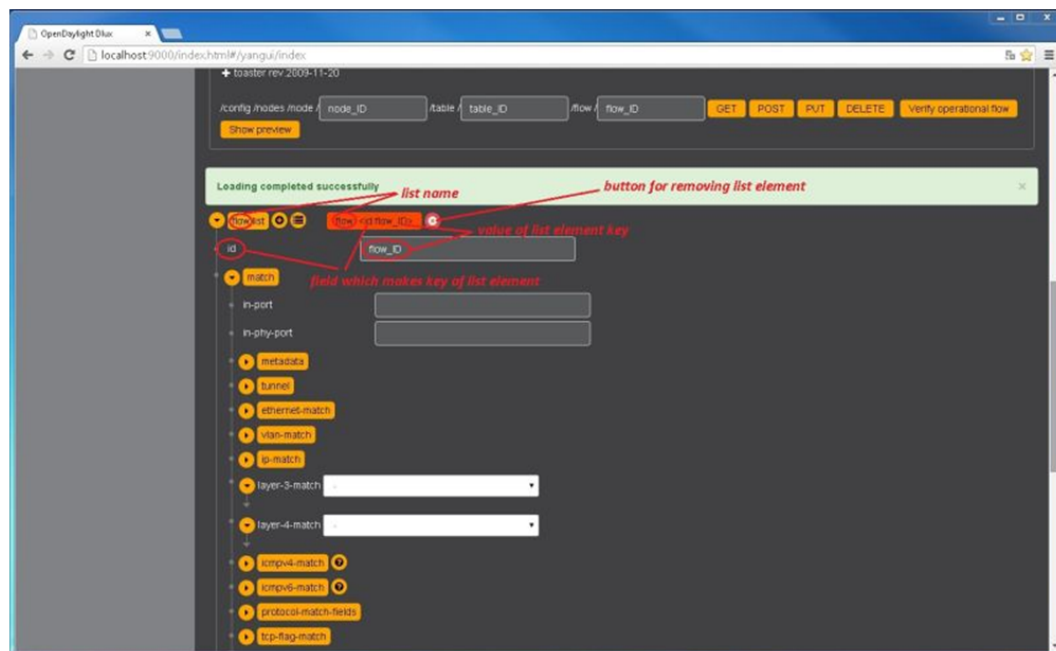
You must specify the xpath for all these operations. This path is displayed in the same row before buttons and it may include text inputs for specific path element identifiers.

Figure 2.4. 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.

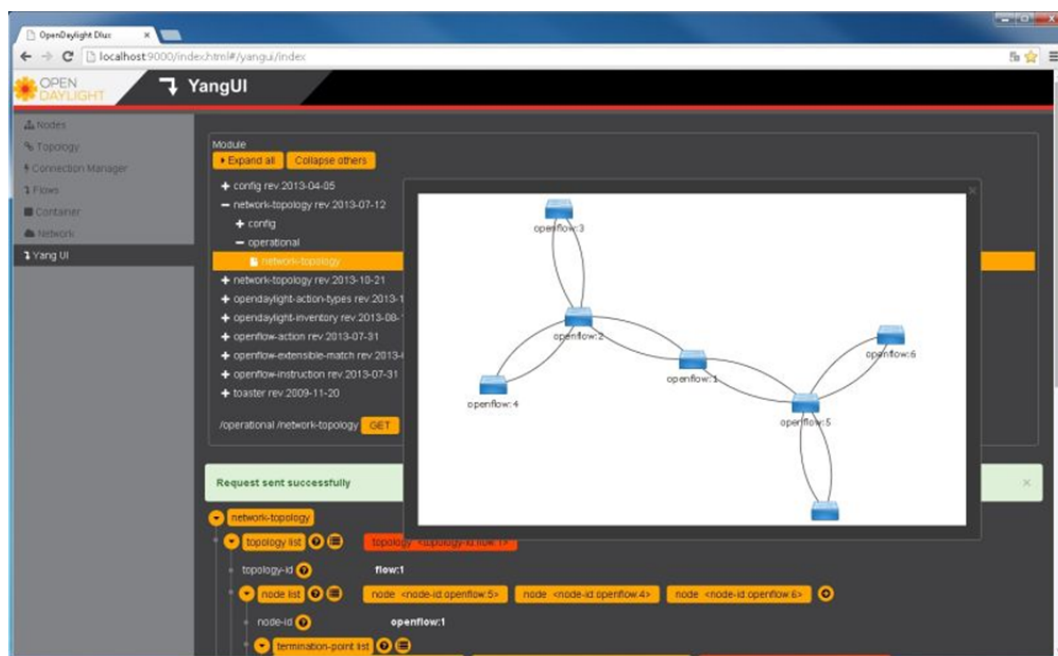
Figure 2.5. Yang UI API Specification

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:

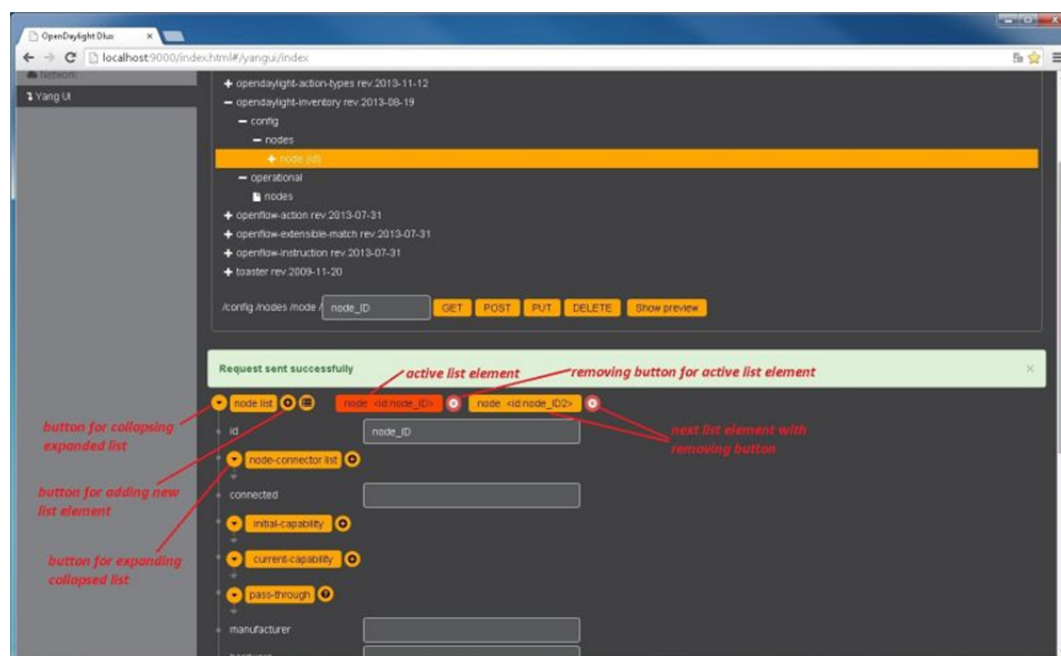
1. Select subAPI network-topology <topology revision number> == > operational == > network-topology.
2. Get data from OpenDaylight by clicking on the "GET" button.
3. Click **Display Topology**.

Figure 2.6. DLUX Yang Topology

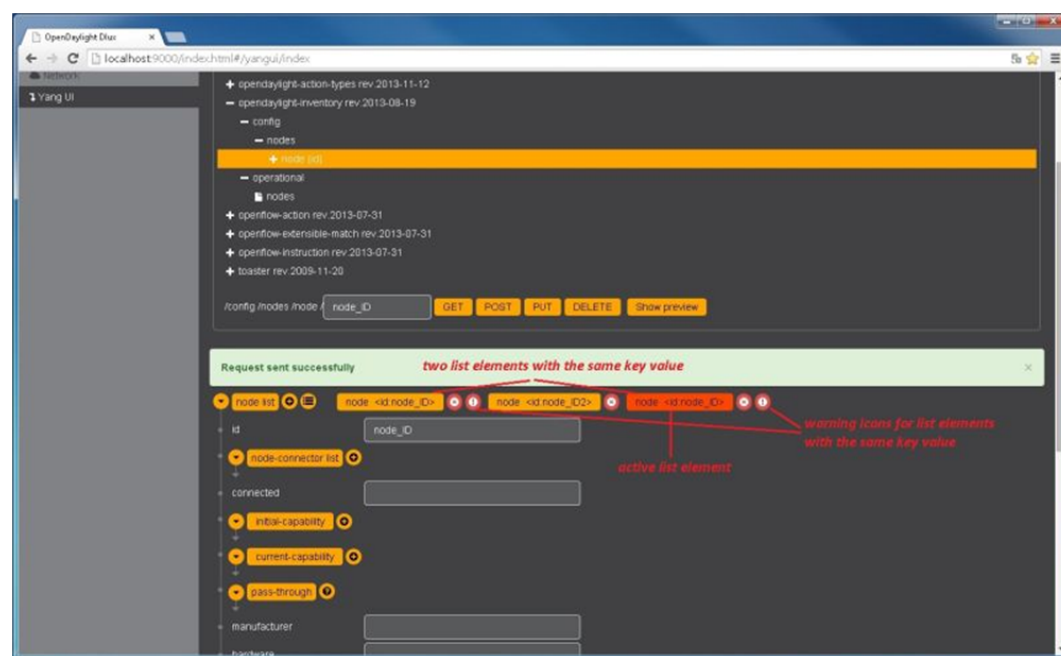
Configuring List Elements on the Yang UI

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.

Figure 2.7. DLUX List Elements

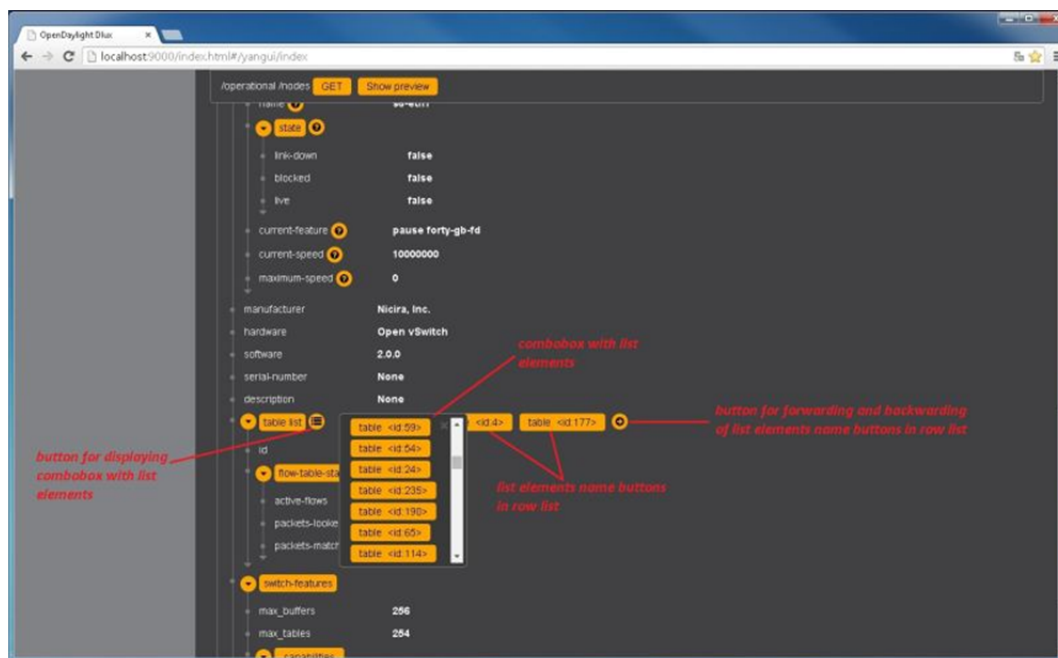
3. In the YANG-based data store all elements of a list must have a unique key. If you try to assign two or more elements the same key, a warning icon ! is displayed near their name buttons.

Figure 2.8. DLUX List Warnings

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.

Figure 2.9. DLUX List Button1



3. Running XSQL Console Commands and Queries

Table of Contents

XSQL Overview	15
Installing XSQL	15
XSQL Console Commands	15
XSQL Queries	16

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-xsq1
```

XSQL Console Commands

To enter a command in the XSQL console, structure the command as follows: **odl:xsq1** *<XSQL command>*

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

Table 3.1. 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 tocsv 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:

Table 3.2. 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.

4. Setting Up Clustering

Table of Contents

Clustering Overview	18
Single Node Clustering	18
Multiple Node Clustering	19

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 many 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.
- **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.

Single Node Clustering

To enable clustering on a single instance of OpenDaylight, perform the following steps:

1. Download, unzip, and run the OpenDaylight distribution
2. Install the clustering feature:

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



Note

This will enable the cluster-ready version of the MD-SAL data store, but will not actually create a cluster of multiple instances. The result is that you will get data persistence, but not the scaling or high availability advantages.

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.

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:
 - a. 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.

- b. 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"]
```

- c. 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.

- d. 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 [below](#).

5. Move into the <karaf-distribution-directory>/bin directory.

6. Run the following command:

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

7. 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"]

    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"
            ]
        }
    ]
}
]
```

5. Security Considerations

Table of Contents

Overview of OpenDaylight Security	24
OpenDaylight Security Resources	25
Deployment Recommendations	25
Securing OSGi bundles	26
Securing the Karaf container	26
Securing Southbound Plugins	27
Securing OpenDaylight using AAA	27
Security Considerations for Clustering	28

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 <http://karaf.apache.org/manual/latest/users-guide/monitoring.html>.
- For remote SSH access configuration, refer to <http://karaf.apache.org/manual/latest/users-guide/remote.html>.
- For WebConsole access, refer to <http://karaf.apache.org/manual/latest/users-guide/webconsole.html>.
- For Karaf security features, refer to <http://karaf.apache.org/manual/latest/developers-guide/security-framework.html>.

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.

Part II. Project-specific Installation Guides

Table of Contents

6. OpFlex agent-ovs Install Guide	31
Required Packages	31
Host Networking Configuration	31
OVS Bridge Configuration	32
Agent Configuration	33
7. OVSDB OpenStack Installation Guide	36
Overview	36
Preparing for Installation	36
Installing OVSDB OpenStack	36
Verifying your Installation	37
Uninstalling OVSDB OpenStack	37
8. OVSDB Service Function Chaining Installation Guide	38
Overview	38
Preparing for Installation	38
Installing OVSDB Service Function Chaining	38
Verifying your Installation	38
Uninstalling OVSDB Service Function Chaining	38
9. OVSDB NetVirt Hardware VTEP Installation Guide	40
Overview	40
Preparing for Installation	40
Installing OVSDB NetVirt Hardware VTEP	40
Verifying your Installation	40
Uninstalling OVSDB NetVirt Hardware VTEP	40
10. TSDR H2 Default Datastore Installation Guide	42
Overview	42
Pre Requisites for Installing TSDR with default H2 datastore	42
Preparing for Installation	42
Installing TSDR with default H2 datastore	42
Verifying your Installation	43
Post Installation Configuration	43
Upgrading From a Previous Release	43
Uninstalling TSDR with default H2 datastore	43
11. TSDR HBase Data Store Installation Guide	44
Overview	44
Prerequisites for Installing TSDR HBase Data Store	45
Preparing for Installation	45
Installing TSDR HBase Data Store	45
Verifying your Installation	46
Post Installation Configuration	46
Upgrading From a Previous Release	46
Uninstalling HBase Data Store	46
12. VTN Installation Guide	48
Overview	48
Preparing for Installation	49
Installing VTN	49
Verifying your Installation	50
Uninstalling VTN	51

6. OpFlex agent-ovs Install Guide

Table of Contents

Required Packages	31
Host Networking Configuration	31
OVS Bridge Configuration	32
Agent Configuration	33

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.

7. OVSDb OpenStack Installation Guide

Table of Contents

Overview	36
Preparing for Installation	36
Installing OVSDb OpenStack	36
Verifying your Installation	37
Uninstalling OVSDb OpenStack	37

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 [Getting and Installing OpenDaylight](#).

Installing OVSDb OpenStack

Install the required features with the following command:

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

Sample output from the Karaf console

```
opendaylight-user@root>feature:list -i | grep ovsdb
odl-ovsdb-southbound-api          | 1.1.0-SNAPSHOT | x          | odl-
ovsdb-southbound-1.1.0-SNAPSHOT
OpenDaylight :: southbound :: api
odl-ovsdb-southbound-impl         | 1.1.0-SNAPSHOT | x          | odl-
ovsdb-southbound-1.1.0-SNAPSHOT
OpenDaylight :: southbound :: impl
odl-ovsdb-southbound-impl-rest    | 1.1.0-SNAPSHOT | x          | odl-
ovsdb-southbound-1.1.0-SNAPSHOT
OpenDaylight :: southbound :: impl :: REST
odl-ovsdb-southbound-impl-ui      | 1.1.0-SNAPSHOT | x          | odl-
ovsdb-southbound-1.1.0-SNAPSHOT
OpenDaylight :: southbound :: impl :: UI
```

```
odl-ovsdb-openstack | 1.1.0-SNAPSHOT | x | ovsdb-1.1.0-SNAPSHOT
OpenDaylight :: OVSDB :: OpenStack Network Virtual
```

Verifying your Installation

To verify that the installation was successful, use the following command in karaf and check that there are no errors logs relating to odl-ovsdb-openstack.

```
log:display
```

Troubleshooting

There is no easy way to troubleshoot an installation of odl-ovsdb-openstack. Perhaps a combination of `log:display | grep -i ovsdb` in karaf, Open vSwitch commands (`ovs-vsctl`) and OpenStack logs will be useful but will not explain everything.

Uninstalling OVSDB OpenStack

1. Shutdown the karaf instance:

```
system:shutdown
```

2. Remove what is in the /data folder of the OpenDaylight Distribution.

8. OVSDB Service Function Chaining Installation Guide

Table of Contents

Overview	38
Preparing for Installation	38
Installing OVSDB Service Function Chaining	38
Verifying your Installation	38
Uninstalling OVSDB Service Function Chaining	38

Overview

TBD

Preparing for Installation

Follow the instructions in [Getting and Installing OpenDaylight](#).

Installing OVSDB Service Function Chaining

Install the required features with the following command:

```
feature:install odl-ovsdb-sfc-ui
```

Sample output from the Karaf console

TBD

Verifying your Installation

To verify that the installation was successful, use the following command in karaf and check that there are no error logs relating to odl-ovsdb-sfc

```
log:display
```

Troubleshooting

TBD

Uninstalling OVSDB Service Function Chaining

1. Shutdown the karaf instance:

```
system:shutdown
```

2. Remove what is in the /data folder of the OpenDaylight Distribution.

9. OVSDb NetVirt Hardware VTEP Installation Guide

Table of Contents

Overview	40
Preparing for Installation	40
Installing OVSDb NetVirt Hardware VTEP	40
Verifying your Installation	40
Uninstalling OVSDb NetVirt Hardware VTEP	40

Overview

TBD

Preparing for Installation

Follow the instructions in [Getting and Installing OpenDaylight](#).

Installing OVSDb NetVirt Hardware VTEP

Install the required features with the following command:

```
feature:install odl-ovsdb-netvirt-hwvtep
```

Sample output from the Karaf console

TBD

Verifying your Installation

To verify that the installation was successful, use the following command in karaf and check that there are no error logs relating to odl-ovsdb-netvirt-hwvtep

```
log:display
```

Troubleshooting

TBD

Uninstalling OVSDb NetVirt Hardware VTEP

1. Shutdown the karaf instance:

```
system:shutdown
```

2. Remove what is in the /data folder of the OpenDaylight Distribution.

10. TSDR H2 Default Datastore Installation Guide

Table of Contents

Overview	42
Pre Requisites for Installing TSDR with default H2 datastore	42
Preparing for Installation	42
Installing TSDR with default H2 datastore	42
Verifying your Installation	43
Post Installation Configuration	43
Upgrading From a Previous Release	43
Uninstalling TSDR with default H2 datastore	43

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 the default JPA (H2) Datastore. TSDR is new functionality added in OpenDaylight in Lithium Release.

Overview

In Lithium Release the time series data records of OpenFlow statistics are collected periodically and stored in a persistent store. For non-production usage, the bundled default JPA based datastore (H2) is utilized based on odl-tdsr-all feature installation. The TSDR records get persisted in H2 store in <install folder>/tsdr/ folder by default.

Pre Requisites for Installing TSDR with default H2 datastore

There are no additional pre-requisites for TSDR based on default datastore

Preparing for Installation

No additional steps required for preparation of installing TSDR feature

Installing TSDR with default H2 datastore

Once OpenDaylight distribution is up, from karaf console install the TSDR feature with default datastore (JPA based datastore H2 store used) can be installed by

```
feature:install odl-tdsr-all
```

This will install all dependency features (and can take sometime) before returning control to the console.

Verifying your Installation

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

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

Troubleshooting

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

Post Installation Configuration

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 H2 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

Upgrading From a Previous Release

Lithium being the first release supporting TSDR functionality, only fresh installation is possible. However if you want to move to production usage by enabling the store HBase for TSDR usage, you can do it by uninstalling the TSDR with default H2 datastore, restarting the Karaf container and then enabling the TSDR with HBase store as documented in tsdr-hbase-install.doc

Uninstalling TSDR with default H2 datastore

To uninstall the TSDR functionality with the default store, you need to do the following from karaf console * feature:uninstall odl-tsdr-all * feature:uninstall odl-tsdr-core * feature:uninstall odl-tsdr-H2-persistence

Its recommended to restart the Karaf container after uninstallation of the TSDR functionality with the default store

11. TSDR HBase Data Store Installation Guide

Table of Contents

Overview	44
Prerequisites for Installing TSDR HBase Data Store	45
Preparing for Installation	45
Installing TSDR HBase Data Store	45
Verifying your Installation	46
Post Installation Configuration	46
Upgrading From a Previous Release	46
Uninstalling HBase Data Store	46

This document is for the user to install the artifacts that are needed for using HBase Data Store in Time Series Data Repository, which is a new feature available in OpenDaylight Lithium release.

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. It contains the following services and components:

- Data Collection Service
- Data Storage Service
- TSDR Persistence Layer with data stores as plugins
- TSDR Data Stores
- Data Query Service
- Data Aggregation Service
- Data Purging Service

Data Collection Service handles the collection of time series data into TSDR and hands it over to Data Storage Service. Data Storage Service stores the data into TSDR through TSDR Persistence Layer. TSDR Persistence Layer provides generic Service APIs allowing various data stores to be plugged in. Data Aggregation Service aggregates time series fine-grained raw data into course-grained roll-up data to control the size of the data. Data Purging Service periodically purges both fine-grained raw data and course-grained aggregated data according to user-defined schedules.

In Lithium, we implemented Data Collection Service, Data Storage Service, TSDR Persistence Layer, TSDR HBase Data Store, and TSDR H2 Data Store. Among these services and components, time series data is communicated using a common TSDR data model, which is designed and implemented for the abstraction of the time series data commonalities.

With these functions, TSDR will be able to collect the data from the data sources and store them into one of the TSDR data stores: either HBase Data Store or H2 Data Store. We also provided a simple query command from Karaf console for the user to retrieve TSDR data from the data stores.

A future release will contain Data Aggregation service, Data Purging Service, and a full-fledged Data Query Service with Norghbound APIs.

Prerequisites for Installing TSDR HBase Data Store

The hardware requirements are the same as those for standard ODL controller installation.

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

- The supported operating system for TSDR HBase Data Store is Linux. We do not support TSDR HBase Data Store on Windows.
- Besides the software that ODL requires, we also require HBase database running on top of Hadoop single node.

Preparing for Installation

Download HBase (version number to be finalized) from the following website.

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

Installing TSDR HBase Data Store

Installing TSDR HBase Data Store contains two steps:

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

This installation guide will only cover the first step. For installing TSDR HBase Data Store features, please refer to TSDR HBase Data Store User Guide.

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

- Create a folder in Linux operating system for the HBase server.

For example, create an hbase directory under /usr/lib:

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

- Unzip the downloaded HBase server tar file.

Run the following command to unzip the installation package:

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

- 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 experience 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>
```

Verifying your Installation

After the HBase server is properly installed, start hbase server and hbase shell.

- a. start hbase server

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

- b. start hbase shell

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

Post Installation Configuration

Please refer to HBase Data Store User Guide.

Upgrading From a Previous Release

Lithium is the first release of TSDR. Upgrading is not applicable for TSDR Lithium release.

Uninstalling HBase Data Store

To uninstall TSDR HBase Data Store,

- a. stop hbase server

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

- b. remove the file directory that contains the HBase server installation.

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

12. VTN Installation Guide

Table of Contents

Overview	48
Preparing for Installation	49
Installing VTN	49
Verifying your Installation	50
Uninstalling VTN	51

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 Controller 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 ODL controller. VTN Manager is implemented as one plugin to the OpenDaylight controller. 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 ODC 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 a user to use the VTN Virtualization. It interacts with VTN Manager plugin to implement the user configuration. It is also capable of multiple controller orchestration. It realizes Virtual Tenant Network (VTN) provisioning in OpenDaylight Controllers (ODC). In the OpenDaylight architecture VTN Coordinator is part of the network application, orchestration and services layer. VTN Coordinator has been implemented as an external application to the OpenDaylight controller. This component is responsible for the VTN virtualization. VTN Coordinator will use the REST interface exposed by the VTN Manager to realize the virtual network using the OpenDaylight controller. It uses OpenDaylight APIs (REST) to construct the virtual network in ODCs. It provides REST APIs for northbound VTN applications and supports virtual networks spanning across multiple ODCs by coordinating across ODCs.

Preparing for Installation

VTN Manager

Running the Karaf distribution

Follow the instructions in [Getting and Installing OpenDaylight](#).

VTN Coordinator

- Arrange a physical/virtual server with any one of the supported 64-bit OS environment.
 - RHEL 6 / 7
 - CentOS 6 / 7
 - Fedora 20 / 21 / 22
- Install these packages

```
yum install perl-Digest-SHA uuid libxslt libcurl unixODBC json-c
```

```
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-rest odl-vtn-manager-neutron
```



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 Lithium

```
cd distribution-karaf-0.2.1-Lithium-SR1/externalapps
```

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

```
tar -C/ -jxvf distribution.vtn-coordinator-6.0.0.1-Lithium-SR1-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 i | 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:8282/controller/nb/v2/vtn/  
default/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

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

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