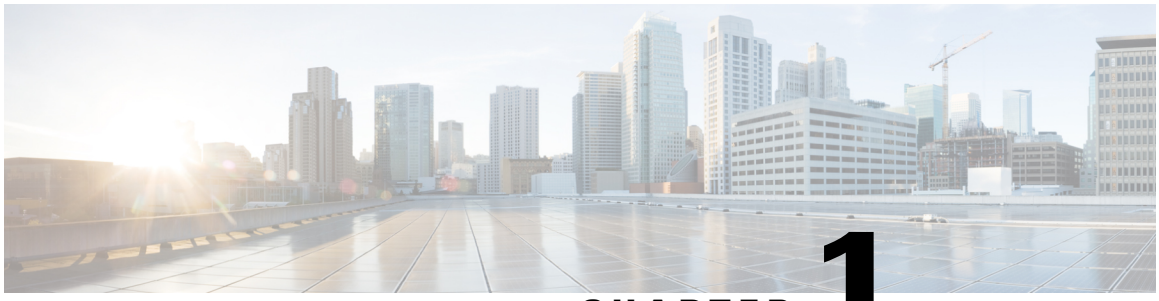
Cisco Open SDN Controller 1.1 Administrator Guide

First Published:

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Open SDN Controller Basics

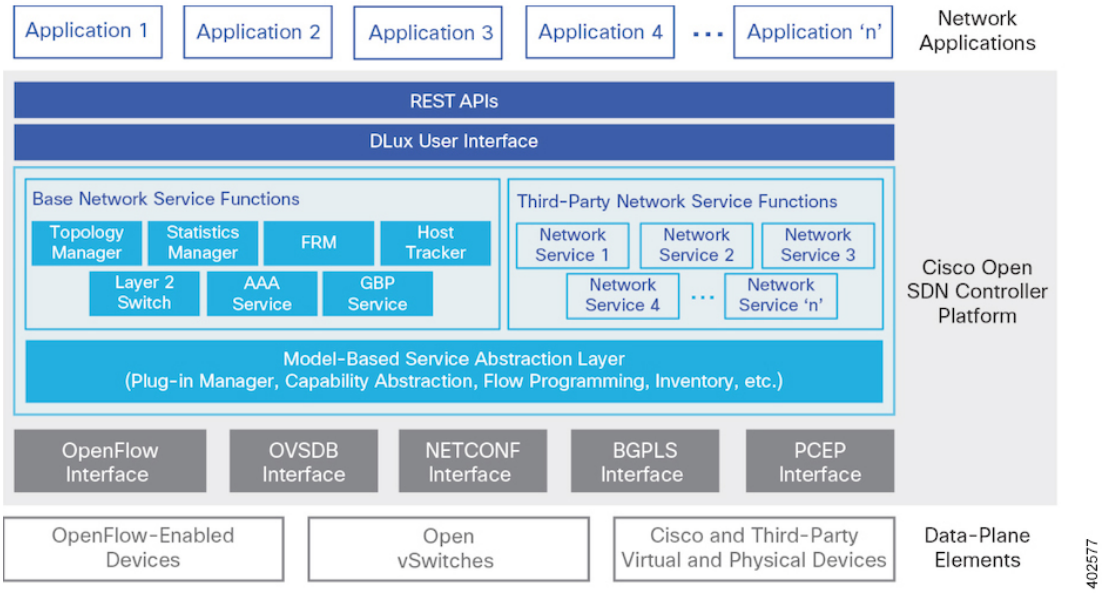
The following topics provide an overview of Open SDN Controller and describe basic information you will need to use this product:

Overview

The Cisco Open SDN Controller is a commercial distribution of OpenDaylight that delivers business agility through the automation of standards-based network infrastructure. Built as a highly scalable software-defined

networking (SDN) application platform, the Open SDN Controller abstracts away the complexity of managing heterogeneous networks to improve service delivery and reduce operating costs.

Figure 1: Open SDN Controller Platform Overview



Key Features

The following table summarizes the key features that Open SDN Controller provides.

Feature	Description
Commercial distribution	Provides a hardened, validated, and supported OpenDaylight distribution.
Clustering	Allows you to configure multiple controller nodes to act as one in order to ensure the controller's continuous operation.
Serviceability	Provides features such as log collection, metrics collection, and system monitoring.
Open Virtual Appliance (OVA) packaging	Enables simplified installation and deployment flexibility.
Cisco DevNet Integration	Provides access to the Open SDN Controller application development environment. For more information, visit developer.cisco.com/site/opensdn.
IP/MPLS and OpenFlow network support	Integrated application support of IP/MPLS and OpenFlow networks

Northbound Representational State Transfer (REST) APIs	Support application integration to the network.
Network services Java APIs	Enable the creation of embedded functions to deliver custom controller capabilities.
Southbound device plug-ins	Connect virtual and physical network elements, supporting heterogeneous network environments.

Logging In

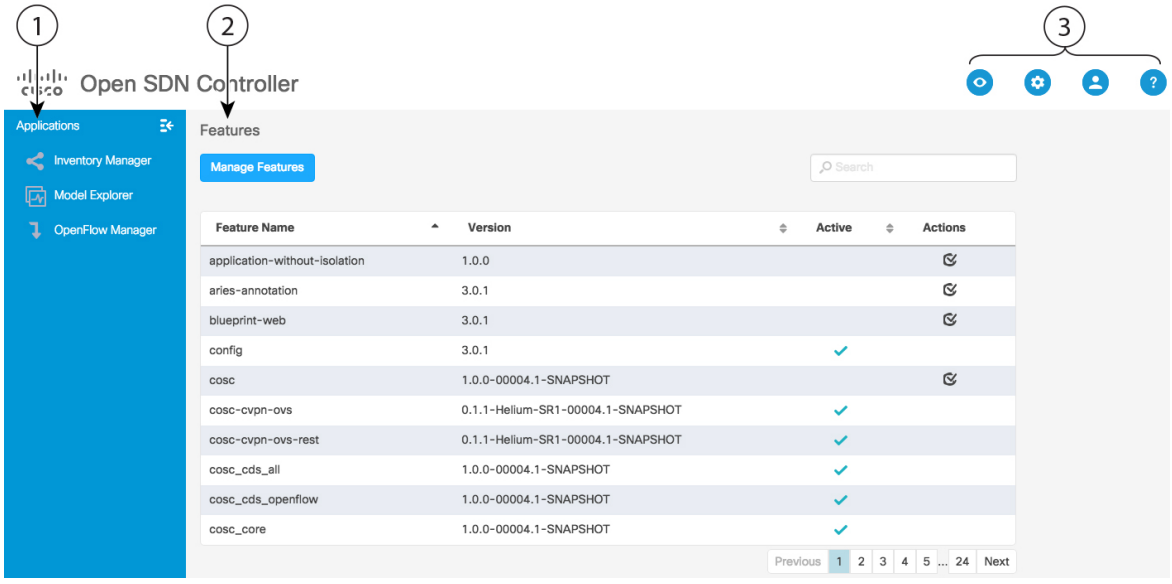
Step 1 In a browser supported by Open SDN Controller, open the following URL: https://<*controller-IP-address*>/#/login

Step 2 In the login screen, enter **admin** as both the username and passphrase and then click **Login**.

Open SDN Controller GUI

The Open SDN Controller GUI is comprised of the following components:

Figure 2: Open SDN Controller GUI Components



Callout	Component	Description

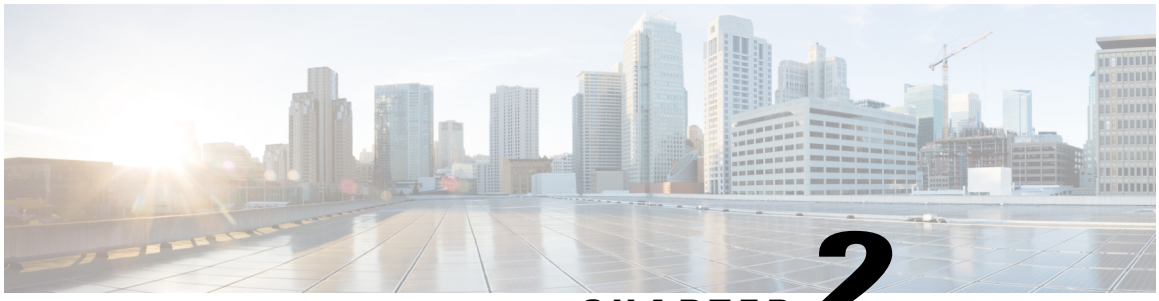
1	Applications pane	Provides a launching point for the applications that Open SDN Controller provides to facilitate the administration of your network: • BGPLS Manager • Inventory Manager • Model Explorer • OpenFlow Manager • PCEP Manager **Note** To minimize or maximize the Applications pane, click the arrow icon located in the top right-hand corner of the pane.
2	Content pane	Displays the content applicable to the application or page that was last opened. • For more information about a particular application or page, see its corresponding topic in this guide. • For a description of the toolbar available in the BGPLS Manager, OpenFlow Manager, and PCEP Manager applications, see Application Toolbar.
3	Main toolbar	Provides the following four menus: • Monitoring menu: From here, you can open the Logs Dashboard, Metrics Dashboard, and Services Status page. See Monitoring Your System for more information. • Management menu: From here, you can open the feature and user management pages. See Managing Your System for more information. • User menu: From here, you can select **Logout** to log out of Open SDN Controller. • Help menu: From here, you can access documentation for the REST APIs that the controller supports. You can also export the latest diagnostic information for your system. See Viewing RESTCONF API Documentation and Exporting Diagnostic Information for more information.

Application Toolbar

The following table describes the features available in the toolbar shared by the BGPLS Manager, OpenFlow Manager, and PCEP Manager applications.

Button	Description
	Click to select one or multiple devices in the topology. To select multiple devices, either hold down the Shift key while you click each device or draw a rectangle around them.

Button	Description
	Click to move the entire topology or particular devices within the topology.
	Click to zoom out the topology.
	Click to zoom in the topology.
	Click to zoom in on a particular area of the topology. Draw a rectangle around the device or devices you want to focus on.
	Click to restore the default topology view and display the entire topology within the content pane.
	When multiple devices are selected, click to group those devices. Once devices are grouped, a single icon depicts all of the devices in that group. To toggle between viewing the group icon and the icons for the individual devices, click the +/- sign.
	Click to view the application in full-screen mode.
	Click to specify how devices are displayed in the topology and the color theme that is used.



CHAPTER 2

Open SDN Controller Applications

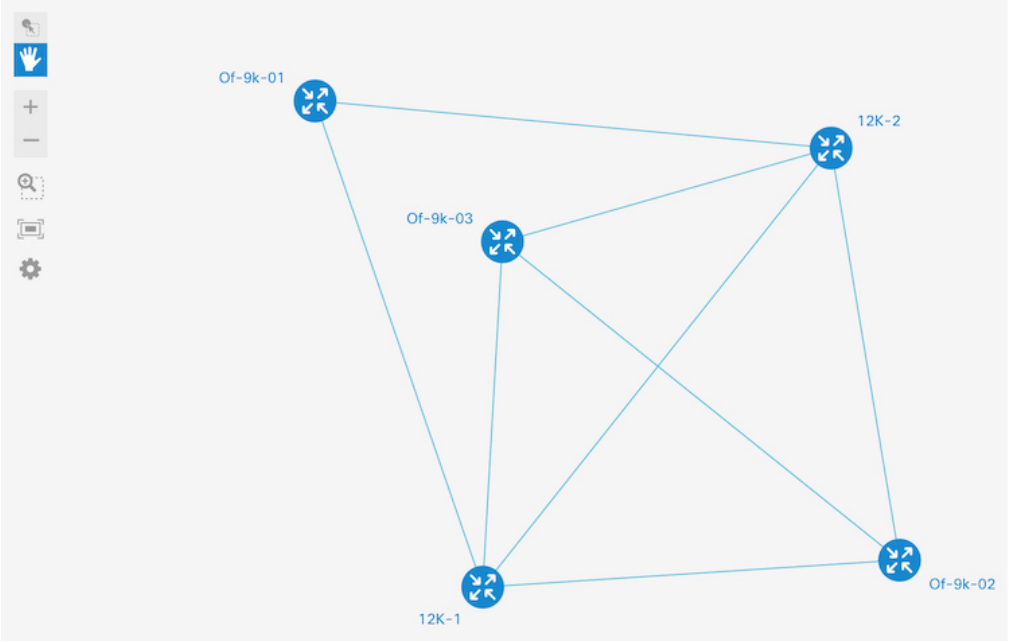
The following topics describe the five applications that Open SDN Controller provides to facilitate the day-to-day administration of your network:

BGPLS Manager

Border Gateway Protocol (BGP) allows you to set up an interdomain routing system that automatically guarantees the loop-free exchange of routing information between autonomous systems. The primary function of a BGP system is to exchange network reachability information with other BGP systems, including information about the list of autonomous system paths. BGP can also be used to exchange other types of network information. For example, BGP Link-State (a BGP extension) enables a router running a link-state routing protocol (such as IS-IS) to communicate the link-state database in a BGP session up to the controller.

From the BGPLS Manager, you can view a topology that maps to the nodes or devices in your network running a link-state routing protocol. You can also view address and interface information for each device, as well as the status of device links.

Figure 3: BGPLS Manager



Note that BGP-enabled devices cannot be added to the topology within BGPLS Manager. To add devices, complete the following procedure described in the Adding BGP Devices topic.

BGPLS Manager Workflow

The following procedure describes a typical workflow you would employ for the BGPLS Manager.

Note Before you proceed, ensure that BGP-LS has been configured properly on both the controller and a BGP-LS speaker in the network. For more information, see the Setting Up BGP-LS and PCEP section in the Open SDN Controller installation guide.

Step 1 Open the BGPLS Manager by selecting **BGPLS Manager** from the Applications pane.
A topology appears in the content pane, displaying the BGP-enabled devices in your network and the links that connect them. In this example, say you want to focus on the device that resides in your San Francisco site and view its BGP information.

Step 2 From the BGPLS Manager toolbar, click **Select**.
Note For a description of the BGPLS Manager toolbar, see Application Toolbar.

Step 3 Click the icon for the San Francisco device.

A popup window opens, displaying basic BGP information for the device such as its IP address and any networks it is set to announce.

Note To quickly determine a device's neighbor devices, place your cursor over that device's icon in the topology. The icons for any non-neighboring devices are dimmed.

Step 4 Click the links connected to the device to view traffic information and identify any links with higher than normal traffic, which could indicate that a problem exists.

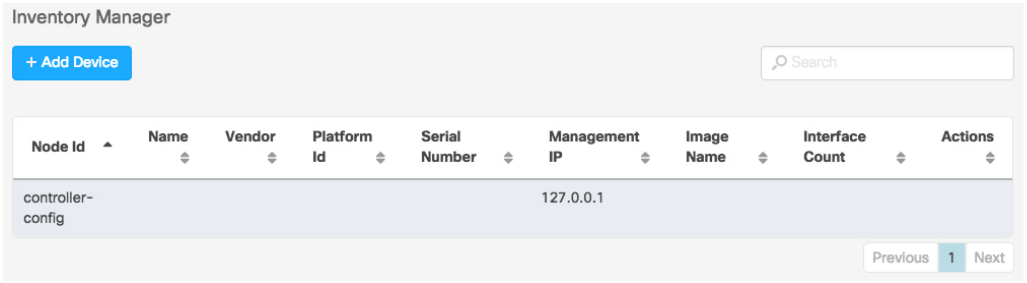
Note See Creating LSPs for a description of how to create links between devices.

Inventory Manager

From the Inventory Manager, you can view summary information for both the devices that Open SDN Controller manages and the interfaces that are configured on those devices. Note the following:

- You are only able to edit or delete NETCONF devices from the Inventory Manager. For a description of how to add devices, see Mounting NETCONF Devices to the Controller.
- Any non-operational NETCONF devices will be indicated by red text.
- After a login username and password have been set on a NETCONF device, you can then change these values directly from the Inventory Manager.

Figure 4: Inventory Manager Application



Inventory Manager Workflow

The following procedure describes a typical workflow you would employ for the Inventory Manager.

Step 1 Open the Inventory Manager by selecting **Inventory Manager** from the Applications pane.
When you first open the Inventory Manager, you see a listing of every managed device in your system, as well as information such as its node ID, serial number, and interface count.

Note If you only want to view information for a specific device, enter its name in the Search field.

Step 2 Manage the devices in your system:

- To add a device, see Adding a Device.
- To edit the settings for a device, see Editing a Device.
- To delete a device, see Deleting a Device.

Step 3 View information for the interfaces configured on a device:

a) Open the Interface Details for *node-ID* page by either clicking anywhere in that device's table row or clicking its interface count value.
This page lists the name, port number, and MAC address for every interface configured on that device.

b) Open the Interface Statistics for *node-ID* page by clicking **Statistics**.
This page provides data transfer statistics such as the number of packets dropped, transmission errors, and collisions.

c) Investigate any interfaces that have higher than normal error numbers.

Note If no devices are connected to the controller, you should still see *controller-config* listed in the first row of the Inventory Manager table. If this entry is not visible, this indicates that the controller inventory APIs are not functioning properly. To confirm this, open the browser developer console. After the page refreshes, you should see either an HTTP 404 or 500 error for `API controller/restconf/operational/opendaylight-inventory:nodes`.

Adding a Device

Step 1 With the Inventory Manager already open, click the Config Nodes tab.

Step 2 Click **Add Device** to open the Add Device dialog box.

Step 3 Specify the following information for the new device:

- Node ID
- IP address
- Port number
- Username and passphrase required to log into the device

Take care when you specify a device's node ID, IP address, and port number because you will not be able to change these settings once the device has been added.

Step 4 Click **Save**.
The device should now be listed in the Config Nodes table.

Editing a Device

Step 1 Select either the Operational Nodes or Config Nodes tab and locate the device you want to edit.

Step 2 From the Actions column, click the device's Edit icon.

Step 3 Make the necessary changes to the username and passphrase required to log into that device and then click **Save**.

Deleting a Device

Step 1 Select either the Operational Nodes or Config Nodes tab and locate the device you want to delete.

Step 2 From the Actions column, click the device's Delete icon.

Step 3 Click **Delete** to confirm the deletion of that device.

Model Explorer

YANG is a data modeling language that models NETCONF configuration data, state data, remote procedure calls (RPCs), and notifications. Open SDN Controller uses YANG models to structure this data hierarchically into modules and submodules and render REST APIs at runtime in the Model Explorer. From here, you can access your network's configuration and state data via REST API methods such as GET and PUT.

To open the Model Explorer, select **Model Explorer** from the Applications pane.

Before you use the Model Explorer, we recommend that you view the documentation available for the REST APIs that Open SDN Controller supports to better understand their syntax and usage. See Viewing RESTCONF API Documentation for more information.

Viewing RESTCONF API Documentation

Open SDN Controller supports a number of RESTCONF APIs. To access documentation that provides usage information for these APIs:

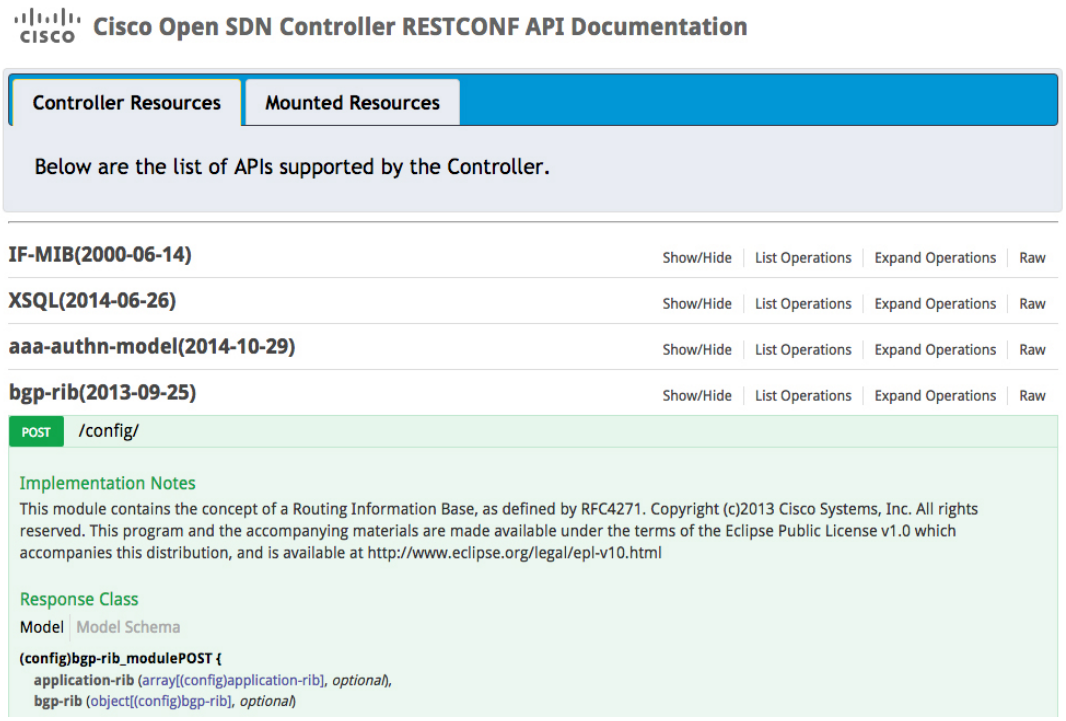
Step 1 From the Help menu, select **API Documentation**.
A browser page opens, displaying all of the APIs that the controller supports.

Note By default, the Controller Resources tab is selected when you first access the API documentation. If you want to view information for the APIs supported by a mounted NETCONF-enabled device, click the Mounted Resources tab.

Step 2 Locate the API you want to view usage information for and click **Expand Operations**.
All of the HTTP methods that the API supports are displayed.

Step 3 To view usage information for a particular method, click its corresponding button (see the following screenshot for an example).

Figure 5: RESTCONF API Documentation Page



Step 4 (Optional) Test a HTTP method to see what is returned based on the values you specify:

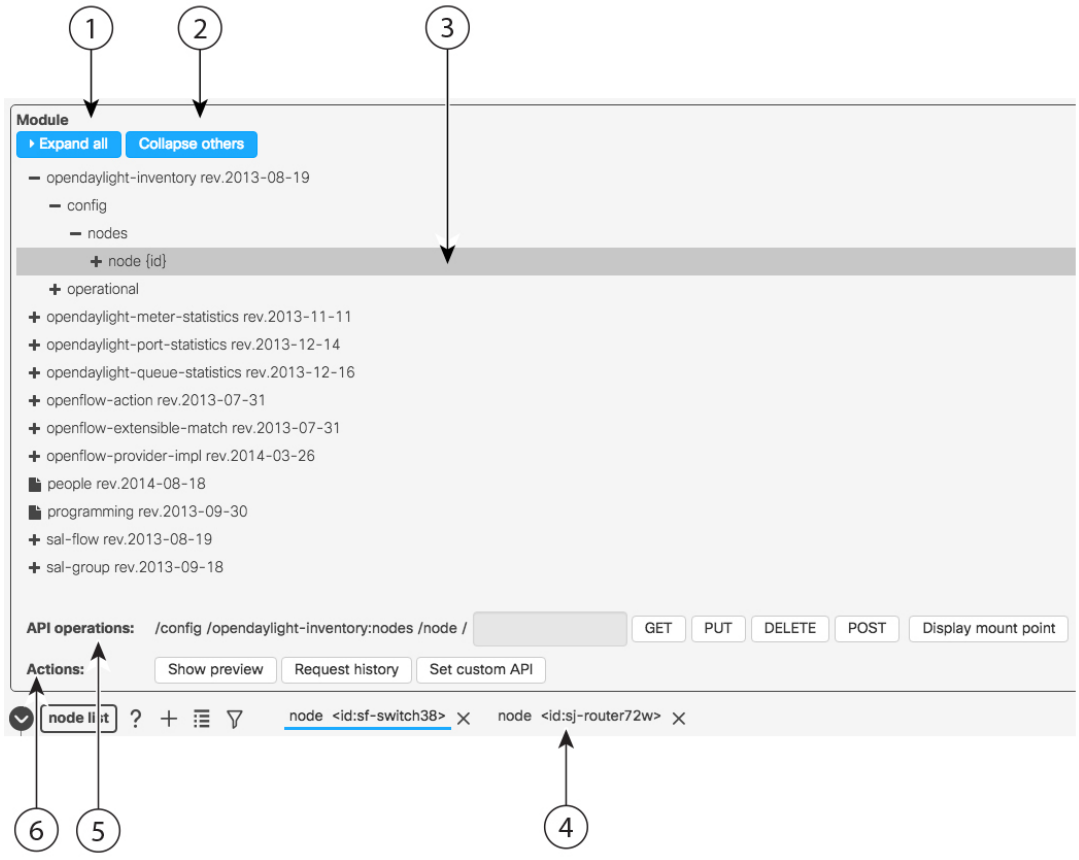
1. If applicable, set the response and parameter content types you want to use.
2. If applicable, enter the parameter values you want to use.
3. Click **Try it out!**

The browser page updates, displaying the corresponding URL for the test request as well as the resulting response, code, and headers.

Navigating the Model Explorer GUI

The following table describes the components that make up the Model Explorer GUI.

Figure 6: Model Explorer GUI Components



Callout	Component	Description
1	Expand all button	Click to expand all of the APIs listed in the Module area and view their elements.
2	Collapse others button	Click to minimize all of the APIs listed in the Module area except for the API that is currently selected and expanded.
3	Module area	Lists every REST API that the controller supports. To work with a particular API, locate it in the list (expanding elements, as needed) and then select the API.
4	API Settings area	Displays the settings configured for the selected API or API operation. From here, you can also add list elements and configure filters.

Callout	Component	Description
5	API Operations field	Lists the corresponding URL for the selected API. Inputs can be filled with data. This field also provides buttons that allow you to execute the operations supported by that API.
6	Actions field	Provides three buttons: • **Show preview**: Click to preview the API path and payload that will be used in an operation you want to execute. • **Request history**: Click to view all of the API method operations that have been executed on the controller. See Viewing Request History for more information. • **Set Custom API**: Click to open the custom API popup window and modify the settings configured for an API. See Modifying API Settings Directly for more information.

Model Explorer Workflow

The following procedure describes a typical workflow you would employ for the Model Explorer.

Step 1 With the Model Explorer open, select an API from the Module area.
What you can do from here depends on whether you selected a config API, an operational API, or an API operation:

- If you selected either a config API or an API operation, the API Settings area updates to display the settings configured for that API or operation. You can then proceed to make the necessary additions or changes. Proceed to Step 2.
- If you selected an operational API, you can view its settings in the API Settings area by clicking **GET** in the API operations field. You will not be able to make any setting changes, so you can either stop here or select another API to work with.

Step 2 Update the settings for the selected API or operation, as needed.
Note the following:

- When you place your cursor over a particular field, a tooltip that indicates the type of value you need to enter (such as a text string or 32-bit unsigned integer) appears.
- Open SDN Controller will indicate any settings you entered incorrectly with an exclamation mark. Place your cursor over the exclamation mark to view a tooltip that describes the error.
- When navigating through an API or operation's elements, click the chevron icons that precede them to expand and collapse the elements.
- For some APIs and operations, you will need to enter path information in the API operations field. Ensure that the information you enter corresponds to the information displayed in the API Settings area.

Step 3 Add list elements to the selected API or operation.
See Adding List Elements for more information.

Step 4 Apply filters to the selected API or operation.
See Managing Filters for more information.

Step 5 (Optional) Click **Show preview** to view the corresponding API path and payload for the operation you want to execute. This feature is useful when you want to copy and paste this information into another application, such as OpenFlow Manager.

Step 6 Execute the appropriate POST, PUT, or DELETE operation.

Adding List Elements

When you view a config API's or operational API's settings, you may see lists that you can modify in the API Settings area. To add elements to a list, complete the following procedure. In this example, we will add two nodes to the opendaylight-inventory API and configure a few settings for each node.

Step 1 From the Module area, select the opendaylight-inventory API.

Step 2 Expand its config and nodes elements and then select the node {id} element.
The node list is now displayed in the API Settings area.

Step 3 Add two nodes to the node list:

a) Click the node list's add list item icon twice.
2 nodes should now be displayed beside the node list.

b) Enter the following settings for each node:
(node 1)

- id: **sf-switch38**
- software: **ios-xr 5.1.2**
- serial-number: **h18si8**

(node 2)

- id: **sj-router72w**
- software: **nx-os 6.0(2)U3(1)**
- serial-number: **z99173**

List Element Operations

The following table describes the operations that are available when you are working with lists in the Model Explorer.

Table 1: Available List Element Operations

Icon	Description
	Click to view a description of an element.
	Click to add a new list element.
	Click to view all of the elements that belong to a list.
	Click to open the filter popup window. See Managing Filters for more information.
	k to view additional list elements so you can select them. Only three elements are displayed in API Settings area at any given time.
	Click to delete a list element.

Managing Filters

The Model Explorer allows you to apply filters to the information it maintains. The following topics describe how to make use of this functionality.

Adding a Filter

Building on the previous example, we will set up a filter for the 2 nodes we added.

Step 1 Click the node list's filter icon to open the filter popup window.
By default, a blank filter is displayed after you first open the filter popup window. You cannot delete it because at least one filter must be displayed in the filter popup window at any given time.

In this example, say you only want to view devices that run Cisco NX-OS software.

Step 2 In the software field, enter **nx-os** and then click **Ok**.
Notice that both nodes are still displayed. This is because = is set as the logical operator (by default), which instructs the Model Explorer to return results that match the value you entered exactly.

Step 3 Reopen the filter popup window and set the software field's logical operator to **contains**.
Only node sj-router72w should be displayed now.

Note the following:

- When multiple filters are configured in the filter popup window, you can switch between them by clicking the appropriate filter button.
- Whenever you click **Ok**, all activated filters will be applied to the selected API or API operation.
- When a filter is configured for a lower-level list, the filter is stored and available to any elements you add to that list, even if you select a different parent object.
- The logical operators that are available for you to choose from will depend on the type of value you need to enter for a particular field.
- When specifying a range for integer values, enter the first and last values in the provided fields.
- To use wildcards:
  1. Set the logical operator to **regEXp** (regular expression).
  2. Enter **[a-z]** for letters and **[0-9]** for numbers.
- For certain elements, you can configure filter settings by selecting the appropriate checkbox in either the API Settings or filter popup window.

Activating and Deactivating a Filter

Step 1 Click a list's filter icon to open the filter popup window.

Step 2 Do one of the following:

- To activate or deactivate an individual filter, click its filter icon to toggle between the two states.
- To deactivate all of the filters that are currently configured, click **Deactivate all** and then click **Ok**.

Step 3 Verify that the filters you have configured are in the correct state by placing your cursor over the list's filter icon and viewing the resulting popup.
A full icon indicates that the filter is activated, whereas an empty icon indicates that the filter is deactivated.

Deleting a Filter

Step 1 Click a list's filter icon to open the filter popup window.

Step 2 Do one of the following:

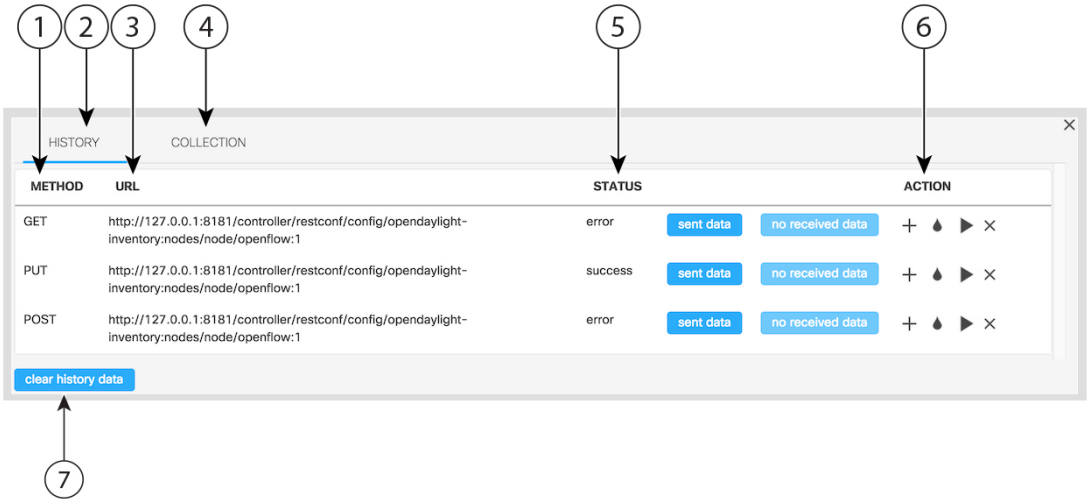
- To delete an individual filter, click its Remove filter icon and then click **Ok**.
- To delete all of the filters that are currently configured, click **Remove all**.

Step 3 Place your cursor over the list's filter icon to verify that the appropriate filter was deleted.
The resulting popup should reflect the changes you made. If you removed all of the filters, the list's filter icon should be empty.

Viewing Request History

The Request history popup window maintains a record of every REST API method operation that has been executed in Open SDN Controller and provides summary information for each operation. The following table describes this window and its components.

Figure 7: Request History Popup Window Components



403211

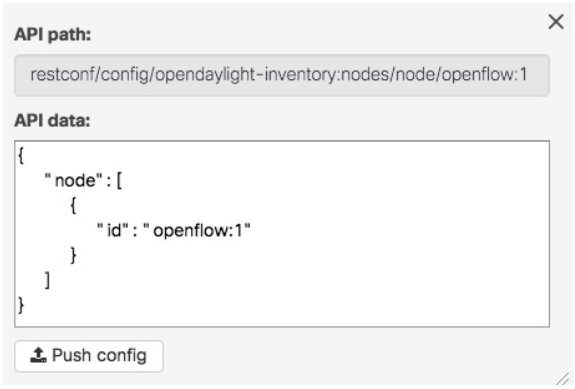
Callout	Component	Description
1	Method column	Indicates the REST API method used for an operation.
2	History tab	Displays the REST API method operations that have been executed on the controller since the last time the Request history popup window's data was cleared.
3	URL column	Indicates the REST API URL used for an operation.
4	Collection tab	Displays the REST API method operations that have been saved by a user for future use.
5	Status column	Indicates whether an operation was executed successfully and whether data was sent or received during the operation.

Callout	Component	Description
6	Action column	Allows you to perform one of the following actions on an operation: • Click to add the operation to the Collection tab. • Click to populate the operation with data. • Click to execute the operation. • Click to delete the operation's entry.
7	clear history data button	Deletes every REST API method operation that is currently listed in the Request history popup window.

Modifying API Settings Directly

When you want to update the settings configured for an API, you would normally do so from the Model Explorer's API Settings area. You also have the option of updating an API's settings directly in the custom API popup window. In the following example, we will configure an API's root node.

Figure 8: Custom API Popup Window



Step 1 From the Actions field, click **Set custom API** to open the custom API popup window.

Step 2 In the API path field, enter the path for the API you want to modify.
In this example, set the API path to
`restconf/config/opendaylight-inventory:nodes/node/openflow:1`.

Step 3 In the API data area, enter the setting changes you want to make.
In this example, enter the following text: **{"node": [{"id": "openflow:1"}]}**

Step 4 Click **Push config**.

OpenFlow Manager

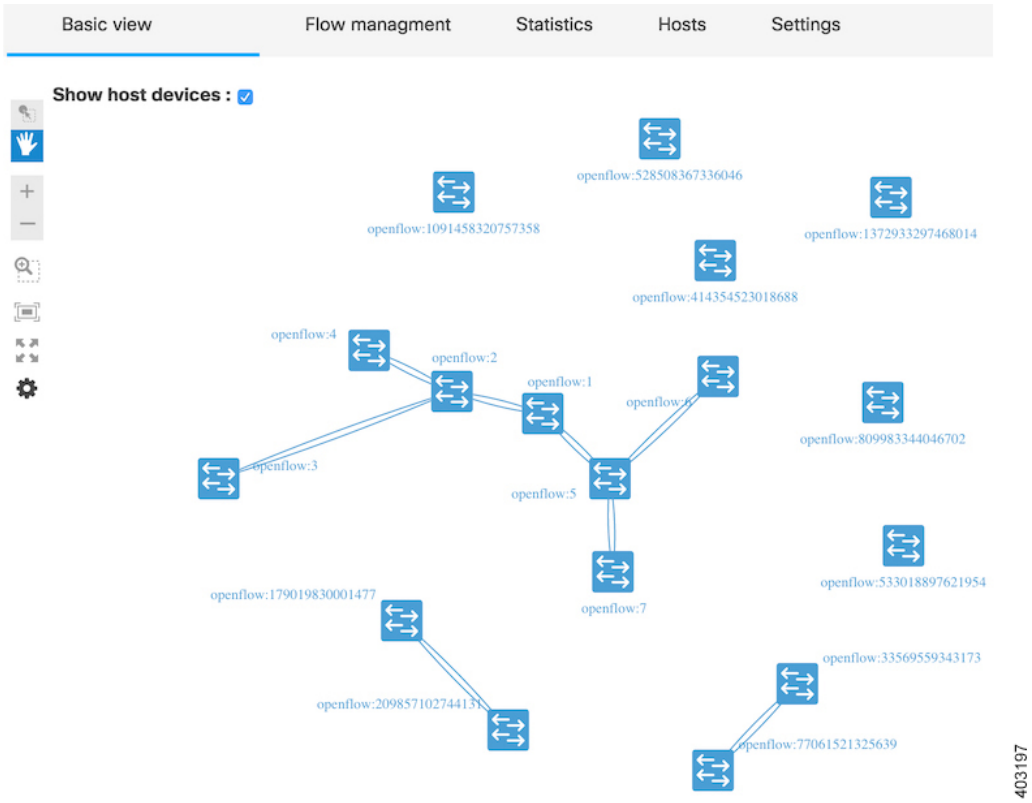
The OpenFlow protocol is based on the concept of an Ethernet switch, with an internal flow-table and standardized interface to allow traffic flows on a switch to be added or removed. The OpenFlow protocol defines the communications channel between the OpenFlow agent and the OpenFlow controller. In an OpenFlow network, the OpenFlow Agent exists on the switch and the OpenFlow controller exists on a server, which is external to the switch. Any network management is either part of the controller or accomplished through the controller.

Open SDN Controller relies on the OpenFlow 1.3 plugin to implement OpenFlow support and provide functionality such as connection creation, session management, state management, and error handling. The plugin, which is installed when you install the controller, allows you to manage the OpenFlow-enabled devices in your network via the OpenFlow Manager. The following components make up this application and are described in more detail in this section:

- Basic View Tab
- Flow Management Tab
- Statistics Tab
- Hosts Tab
- Settings Tab

To open the OpenFlow Manager, select **OpenFlow Manager** from the Applications pane.

Figure 9: OpenFlow Manager Application



Basic View Tab

By default, the Basic View tab is displayed after you open the OpenFlow Manager. The topology provided here maps the OpenFlow-enabled devices in your network and the hosts that are connected to them.

To display hosts, do the following:

1. Ensure that the L2switch feature is enabled on the controller.
2. If you are using Mininet, discover hosts by running the **pingall** command from Mininet.

   Otherwise, proceed to Step 3.
3. With the Basic View tab open, click the **Show host devices** check box.

After selecting a device in the topology, you can manage the flows configured on that device and view the corresponding statistics.

Note To quickly determine a device's neighbor devices, place your cursor over that device's icon in the topology. The icons for any non-neighboring devices are dimmed.

See the following topics for more information:

- For information about adding devices to the topology, see Adding OpenFlow-Enabled Devices.
- For information about configuring secure connections between the controller and devices, see Enabling TLS Support.
- For a description of the OpenFlow Manager toolbar, see Application Toolbar.

Adding OpenFlow-Enabled Devices

Devices cannot be added to the topology within OpenFlow Manager. To add a device, refer to the Open SDN Controller installation guide and complete the procedure specific to the type of device you are adding:

- To add a Cisco ASR 9000 Series router, see Configuring OpenFlow Support on a Cisco ASR 9000 Series Router.
- To add a Cisco Nexus 3000 Series switch, see Configuring OpenFlow Support on a Cisco Nexus 3000 Series Switch.

Flow Management Tab

From this tab, you can perform the following flow management tasks:

- Determine the number of flows associated with each OpenFlow-enabled device in your network.
- View a listing of all the flows that are currently configured.
- Set the deployment mode for a particular device.
- Add, modify, and activate filters to refine the information displayed in the Flows table.
- Add, modify, delete, and reload flows.

Flow Management Tab Workflow

The following procedure describes a typical workflow you would employ for this tab.

Step 1 With the OpenFlow Manager already open, click the Flow Management tab.

Step 2 View the Flow Summary table at the top of the tab, which lists the number of flows (both configured and pending) associated with each OpenFlow-enabled device.

Note You can filter the information provided in this table by entering a device or device type value in the corresponding filter field.

Step 3 Set the deployment mode for the devices listed in the Flow Summary table by clicking the appropriate icon:

- For Proactive mode, click the P icon.
- For Reactive mode, click the R icon.
- For Integrated mode, click the I icon.

For a description of these modes, see Flow Deployment Modes.

Step 4 Scroll to the middle of the tab and manage the filters you want to apply to the Flows table. See Adding a Filter, Modifying a Filter, and Deleting a Filter for more information.

Step 5 Scroll to the bottom of the tab and manage the flows configured on the controller. See Adding a Flow, Modifying a Flow, and Deleting a Flow for more information.

Adding a Filter

To apply a simple filter to the Flows table, enter a value in the filter field that corresponds to a particular column in the Flows table. If you want to create a more robust filter, complete the following procedure.

Step 1 From the Filters table toolbar, click the Filter Management icon.

Step 2 Configure the filter you want to add:

a) In the Filter name field, enter a name for the filter you are creating.
b) From the Device drop-down list, select the device you want to base the filter on.
c) From the General Properties, Match, and Actions lists, select the parameters you want to base the filter on by clicking the appropriate slider buttons.
d) For each parameter you select, enter the value that flows listed in the Flows table should contain.

Step 3 Do one of the following:

- If you want to create an additional filter, click the Create new empty filter icon at the top of the tab and go back to Step 2.
- If you do not want to create an additional filter, proceed to Step 4.

Step 4 Click **Save and Exit**.

Step 5 In the Filters table, confirm that the filter you just created is listed and its check box is selected.

Activating a Filter

Any filters that have been configured will only be applied to the Flows table if they are active.

Step 1 View the heading for the Filters table:

- If the heading reads `Filters inactive`, proceed to Step 2.
- If the heading reads `Filters active`, skip ahead to Step 3.

Step 2 From the Filters table toolbar, click the Activate/Deactivate filters icon.

Step 3 Confirm that the check box for every filter you want to apply to the Flows table is selected.

Modifying a Filter

Step 1 From the Filters table toolbar, click the Filter Management icon.

Step 2 If multiple filters are currently configured, select the filter you want to modify.

Step 3 Make any necessary changes and then click **Save and exit**.

Deleting a Filter

There are two ways to delete a filter:

(From the Filters table)

Locate the filter in the table and click the corresponding Delete icon.

(From the Filters creation/modification tab)

1. From the Filters table toolbar, click the Filter Management icon.
2. At the top of the tab, locate the filter you want to delete and click its Delete icon.

Adding a Flow

Step 1 From the Flows toolbar, click the Flow Management icon.

Step 2 Configure the settings for the flow you want to add:

a) From the device drop-down list, select the source device.

b) From the General Properties, Match, and Actions lists, select the parameters you want to define by clicking the appropriate slider buttons.

Note The Table, ID, and Priority parameters are mandatory and cannot be removed from a flow.

c) Specify the appropriate value for each parameter you selected.

Step 3 Do one of the following:

- If you want to create an additional flow, click the Create new empty flow icon at the top of the tab and go back to Step 2.
- If you do not want to create an additional flow, proceed to Step 4.

Step 4 (Optional) Click **Show Preview** to view the actual code that will be sent to the controller when you submit your request.

Step 5 Click **Send Request**.
If you have configured multiple flows, click **Send All** instead.

Step 6 Click the Reload flows icon to update the Flows table.
The flow you created should now be listed here.

Modifying a Flow

Step 1 From the Flows table, select the check box for the flow you want to modify.

Step 2 Do one of the following:

- Click the Flow Management icon.
- From the Actions column, click the Edit icon.

Step 3 Make any necessary changes and then click **Send Request**.

Note Only the fields you can update will be editable.

Step 4 From the Flows toolbar, click the Reload Flows icon.

Deleting a Flow

There are three ways to delete a flow:

(From the Flows table)

To delete an individual flow:

1. Click its Delete icon in the Actions column.
2. Click **OK** to confirm deletion in the popup window.

To delete multiple flows:

1. Select the check box for every flow you want to delete.
2. From the Flows toolbar, click the Delete icon.
3. Click **OK** to confirm deletion in the popup window.

(From the Flows creation/modification tab)

1. From the Flows toolbar, click the Flow Management icon.
2. At the top of the tab, locate the flow you want to delete and click its Delete icon.

Note the following:

- When you delete a flow that resides in the controller's configuration, it is deleted from the Flows table immediately.
- When you delete a flow that resides on a device, that flow will continue to be displayed in the Flows table until it is removed from the corresponding device.

Statistics Tab

The Statistics tab provides statistics for both the flows configured in your network and the corresponding device ports.

Statistics Tab Workflow

Complete the following procedure to access these statistics.

Step 1 Select **OpenFlow Manager** from the Applications pane. The Basic View tab is open, by default.

Step 2 Do one of the following:

- To view statistics for one or multiple devices, click **Select** from the OpenFlow Manager toolbar and proceed to Step 3.
- To view statistics for all of the OpenFlow-enabled devices in your network, click the Statistics tab and skip ahead to Step 5.

Step 3 Select the device(s) you want to view statistics for.
The page updates, displaying the following buttons:

- Port Stats
- Flow Stats
- Flow Table Stats
- Aggregate Flow Stats
- Queue Stats
- Group Stats
- Meter Stats
- Meter Features Stats

Step 4 Click the button that corresponds to the statistics you want to view and skip ahead to Step 7.

Step 5 From the Statistics drop-down list, select whether you want to view flow (table) or port statistics.
If you select the Table Statistics option, proceed to Step 6. Otherwise, skip ahead to Step 7.

Step 6 From the Type drop-down list, select one of the following options:

- Flow Stats

- Flow Table Stats
- Aggregate Flow Stats
- Queue Stats
- Group Stats
- Meter Stats
- Meter Features Stats

Step 7 Below the statistics table, specify the number of table rows and objects you want to view and then click **Refresh Data**.

Hosts Tab

The Hosts tab provides summary information for the OpenFlow-enabled host devices that Open SDN Controller manages. From here, you can quickly determine things like a host device's ID, attachment point status, and HTS IP address. To specify how many host devices are listed in the Hosts table, select the appropriate value below the table.

Settings Tab

You can configure two settings in the Settings tab: the refresh interval for OpenFlow device statistics and the flow instantiation mode OpenFlow Manager should use.

Step 1 With the OpenFlow Manager already open, click the Settings tab.

Step 2 Configure the following settings:

- Statistics timer-Specifies how often OpenFlow device statistics are refreshed, in seconds.
- Deployment mode-Indicates the flow instantiation mode configured for OpenFlow Manager. There are three modes to choose from:
  - **Proactive**
  - **Reactive**
  - **Integrated**

  For more information on these three modes, see Flow Deployment Modes.

Step 3 Click **Confirm settings** to save your changes to the controller.

Flow Deployment Modes

In the Settings tab, you can specify OpenFlow Manager to use one of three modes when instantiating flows: Reactive Mode, Proactive Mode, or Integrated Mode.

Note the following:

- After you install either the odl-l2switch-switch or odl-openflowplugin-apps feature, the deployment-mode-manager feature is available for use. The functionality provided by this feature depends on which of the two previously mentioned features are installed.
- In the description of each mode, an asterisk indicates that an action is performed only when the odl-l2switch-switch feature is installed.
- Say you have a scenario where either the odl-l2switch-switch or odl-openflowplugin-apps feature is installed and the deployment mode on certain devices has been changed from the default. After you install the other feature, connected devices are updated with the correct flows based on the deployment mode that was set for each device prior to installation.
- Reactive mode is the initial default deployment mode that is automatically applied to all connected devices. To set either Proactive or Integrated mode as the default, update the flow deployment mode configuration file:

  1 Navigate to the directory in which the configuration file resides:

    cd etc/opendaylight/karaf/

  2 Open 42-deployment-mode-manager.xml in a text editor.

  3 Locate the following line and replace `REACTIVE` with the mode you want to configure:

    ```
    <default-deployment-mode>REACTIVE</default-deployment-mode>
    ```

    When specifying a mode, use all uppercase letters.

  4 Save your changes.

- To set a different deployment mode on a particular device, locate the device's entry in the Flow Summary table (in the Flow Management tab) and click the appropriate icon.
- When you change the default deployment mode, that mode is automatically applied only to newly connected devices from that point on. The deployment mode that's already set for devices that connected previously is unchanged.

Reactive Mode

When this mode is selected, OpenFlow Manager forwards a flow's unmatched packets (packets that don't match any entries in a flow table) to the controller, allowing the controller to decide what to do with them. This decision is then stored as a flow entry in the relevant flow table, allowing any packets received for that flow in the future to be processed without controller intervention.

OpenFlow Manager carries out the following actions when Reactive Mode is selected:

1 Punts all packets to the controller.

2 *Punts incoming ARP packets to the controller.

3 *Floods ARP packets.

Proactive Mode

When this mode is selected, OpenFlow Manager pushes all known flows to the network elements that handle forwarding before any traffic is received. Since the flows and their corresponding instructions have already been defined, the controller doesn't need to step in and decide what to do with any unmatched packets (unless that is what you have instructed the controller to do).

OpenFlow Manager carries out the following actions when Proactive Mode is selected:

1. Punts LLDP packets to the controller.
2. Drops remaining packets.
3. *Punts incoming ARP packets to the controller.
4. *Floods ARP packets.

Integrated Mode

This mode is essentially a hybrid of Reactive and Proactive Mode, instructing the controller to figure out what to do with packets that don't match any of the flows that have been defined.

OpenFlow Manager carries out the following actions when Integrated Mode is selected:

1. Punts LLDP packets to the controller.
2. Forces NORMAL routing.
3. *Punts incoming ARP packets to the controller.

Determining the Current Deployment Mode

When you need to determine the deployment mode that is currently set for either the controller or a particular device, you can do so by making one of the following POST requests.

Note the following:

- Before every RESTCONF request you make, you must first generate a security token. See Making RESTCONF Requests for more information.
- You can also determine the deployment mode that is currently set for the controller by viewing OpenFlow Manager's Settings tab.

For the Controller

(URL)

https://token:$token@<*controller-IP-address*>/controller/restconf/operations/deployment-mode:get-deployment-mode

For a Device

(URL)

https://token:$token@<*controller-IP-address*>/controller/restconf/operations/node-deployment-mode:get-node-deployment-mode

(Payload)

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:controller:node-deployment-mode">
   <node
```

```
xmlns:inv="urn:opendaylight:inventory">/inv:nodes/inv:node[inv:id="<OpenFlow-device-ID>"]</node>
</input>
```

Setting the Deployment Mode Manually

Although you can set the deployment mode for the controller and devices from OpenFlow Manager, you also have the option of doing so manually. To do so, make one of the following POST requests, replacing <*deployment-mode*> and <*OpenFlow-device-ID*> with the correct value in the request's payload.

Note Before every RESTCONF request you make, you must first generate a security token. See Making RESTCONF Requests for more information.

For the Controller

(URL)

https://token:$token@<*controller-IP-address*>/controller/restconf/operations/deployment-mode:set-deployment-mode

(Payload)

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:controller:deployment-mode">
<deployment-mode><deployment-mode></deployment-mode>
</input>
```

For a Device

(URL)

https://token:$token@<*controller-IP-address*>/controller/restconf/operations/node-deployment-mode:set-node-deployment-mode

(Payload)

```
<input xmlns="urn:opendaylight:params:xml:ns:yang:controller:node-deployment-mode">
   <node
xmlns:inv="urn:opendaylight:inventory">/inv:nodes/inv:node[inv:id="<OpenFlow-device-ID>"]</node>

   <deployment-mode><deployment-mode></deployment-mode>
</input>
```

PCEP Manager

Path Computation Element Communication Protocol (PCEP) is a TCP-based protocol that defines a set of messages and objects used to manage PCEP sessions and to request and send paths for multi-domain traffic engineering Label Switched Paths (LSPs). From the PCEP Manager, you can create LSPs between the BGP-enabled devices in your network. To open this application, select **PCEP Manager** from the Applications pane.

Note the following:

- The BGPLS Manager and PCEP Manager display the same topology. To add devices to this topology, complete the procedure described in the Adding BGP Devices.
- The list of configured LSPs that is displayed after you click the Establish LSP icon is not updated in real time. It is refreshed only after a change has been made (such as adding or deleting an LSP).
- For a description of the PCEP Manager toolbar, see Application Toolbar.

Before You Get Started

Keep the following information in mind before you use the PCEP Manager.

Requirements

The PCEP Manager assumes that:

- All the BGP-LS/PCEP-enabled routers in your network have a hostname.
- The BGP-Router-ID, MPLS-TE ID, and PCC value configured for each router is the same IP address.
- All router IDs must be reachable from the Open SDN Controller host and vice versa. Specifically, the routers must be able to reply to the controller via a static, default, or dynamic route. **One** of the following must also be true:
  - The controller has a static route to each router loopback address.
  - The controller uses dynamic routing.
  - The controller uses a default route to a node that can reach the router's loopbacks.

Caveats

- The Terminal feature (accessed by right-clicking a device) has been disabled in this release.
- Multipoint links have not been tested.
- A maximum of 50 routers are displayed on the topology at any given time.

Troubleshooting

Problem: No nodes are displayed on the topology.

Solution:

1. Make a GET request, using the following URL—

   https://token:$token@<*controller-IP-address*>/controller/restconf/operational/network-topology:network-topology/topology/example-linkstate-topology

2. In the topology section of the resulting output, find every instance of *router-id* and note the IP address of the corresponding device.

   Every device listed here should be displayed in the topology. If any devices are not displayed, this indicates that they were not configured properly.

Note For a description of how to make RESTCONF requests in Open SDN Controller, see Making RESTCONF Requests.

Problem: When two nodes are selected in the Auto Path tab, a list of the available paths between those nodes is not displayed.

Solution:

1. Check that a PCEP topology is available.
2. If so, check that the loopback IDs for the two nodes are reachable from the controller.
3. If so, verify that the PCE ID and BGP router ID values are the same.

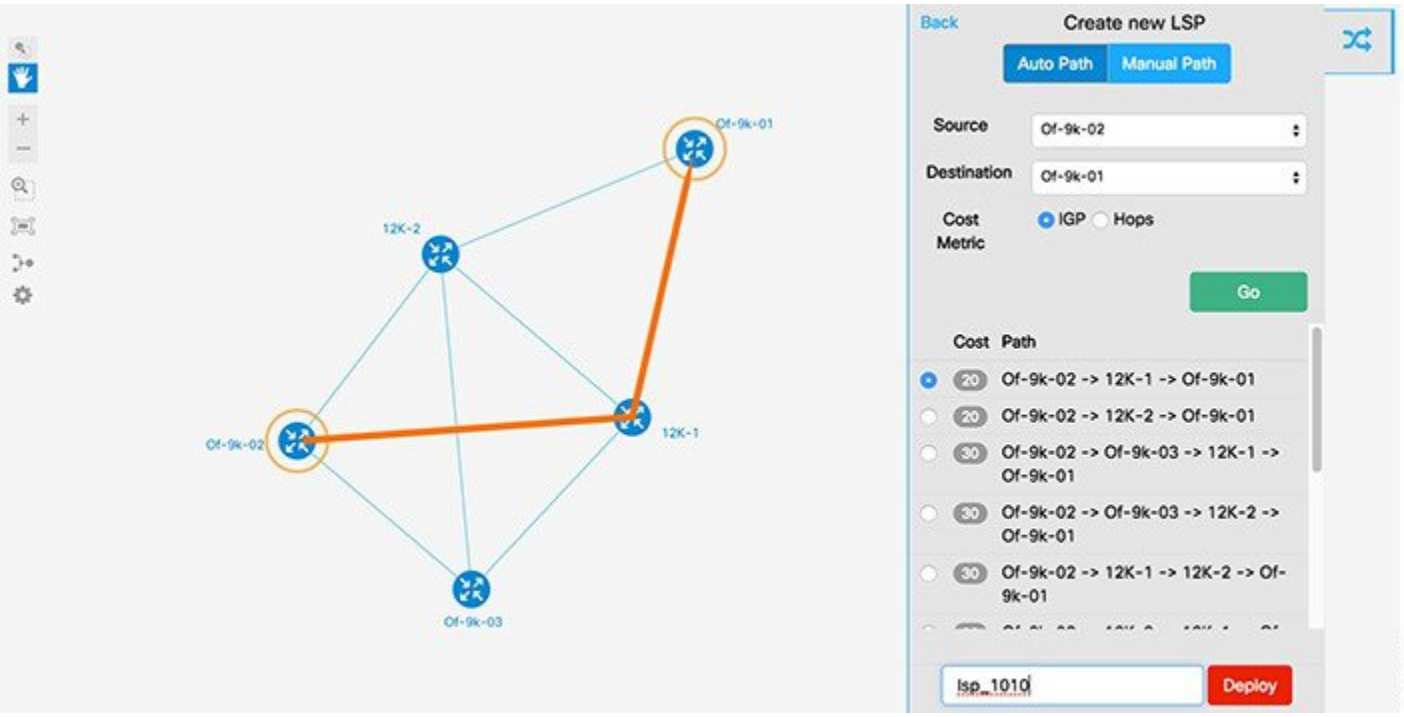
Creating LSPs

You can create three types of LSPs in PCEP Manager: path-based, hop-based, and manual. Complete the procedure for the LSP type you want to create.

Note To quickly determine a device's neighbor devices, place your cursor over that device's icon in the topology. The icons for any non-neighboring devices are dimmed.

Path-Based LSPs

Figure 10: Path-Based LSP Creation Page



Step 1 From the upper right-hand corner of the content pane, click the Establish LSP icon.
A listing of the LSPs that are currently active is displayed.

Step 2 Click **Create New LSP**.
By default, the Auto Path tab and IGP radio button are already selected.

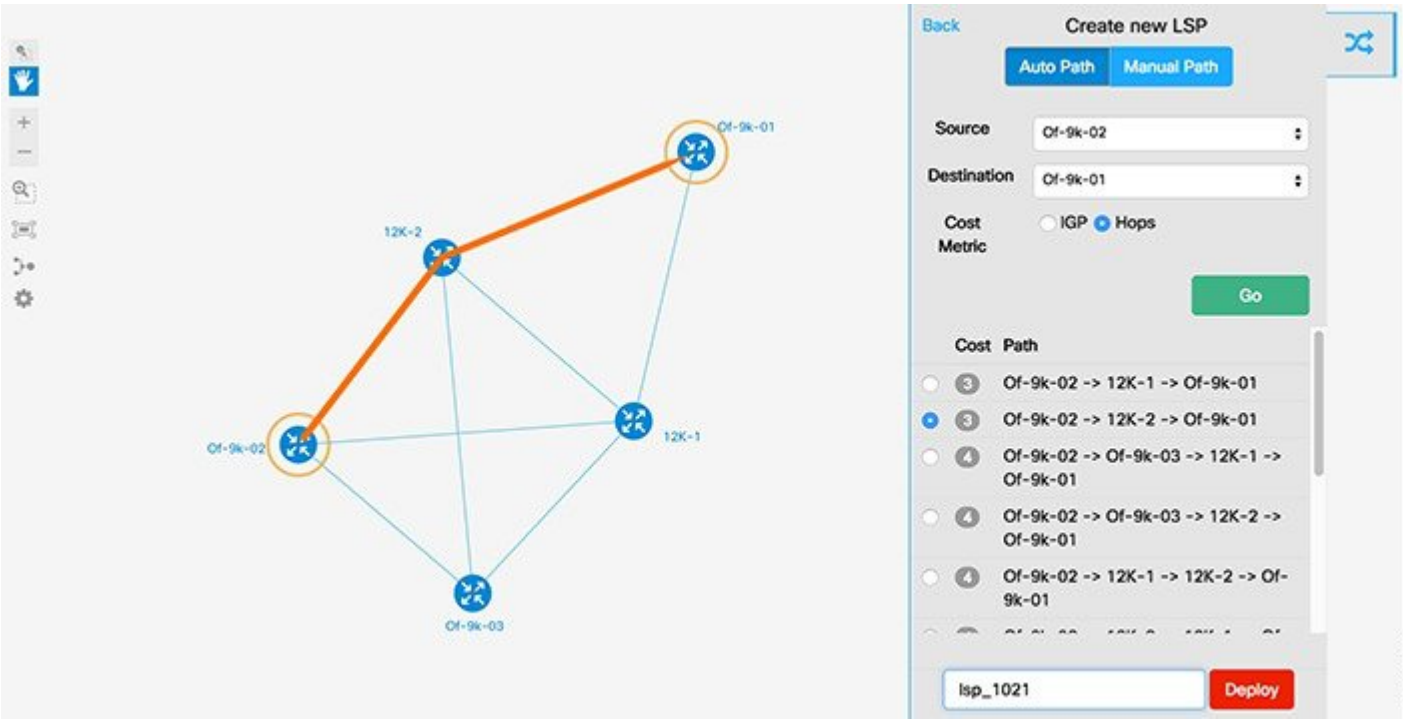
Step 3 Select the source and destination device and then click **Go**.
A listing of the available paths between the two devices is displayed, sorted by IGP metric order.

Step 4 Select the path you want to use.

Step 5 In the LSP Name field, enter a name for the new LSP and then click **Deploy**.

Hop-Based LSPs

Figure 11: Hop-Based LSP Creation Page



Step 1 From the upper right-hand corner of the content pane, click the Establish LSP icon.
A listing of the LSPs that are currently active is displayed.

Step 2 Click **Create New LSP**.
By default, the Auto Path tab and IGP radio button are already selected.

Step 3 Specify the source and destination device.

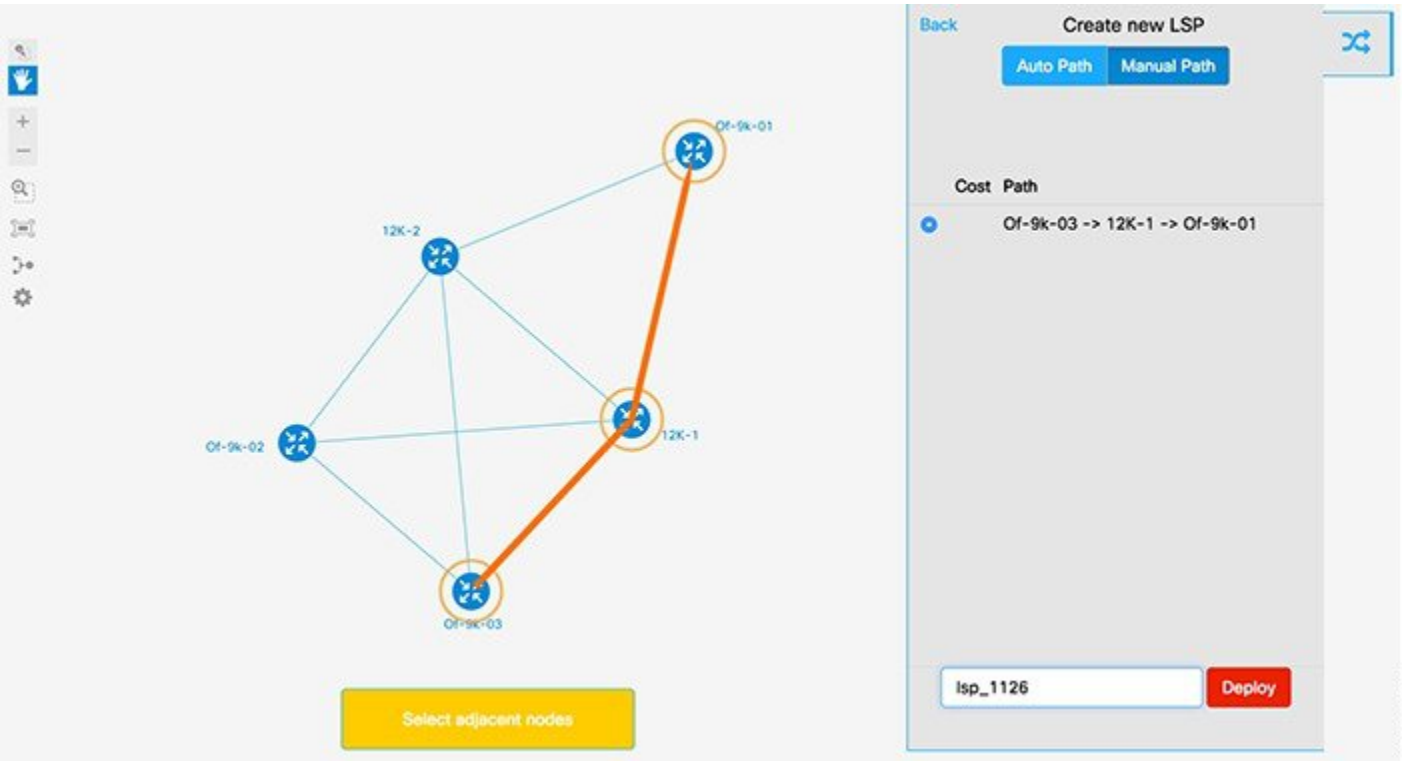
Step 4 Select the Hops radio button and then click **Go**.
A listing of the available paths between the two devices is displayed, along with the cost associated with those paths.

Step 5 Select the path you want to use.

Step 6 In the LSP Name field, enter a name for the new LSP and then click **Deploy**.

Manual LSPs

Figure 12: Manual LSP Creation Page



\

Step 1 From the upper right-hand corner of the content pane, click the Establish LSP icon.
A listing of the LSPs that are currently active is displayed.

Step 2 Click **Create New LSP**.
By default, the Auto Path tab is already selected.

Step 3 Select the Manual Path tab.

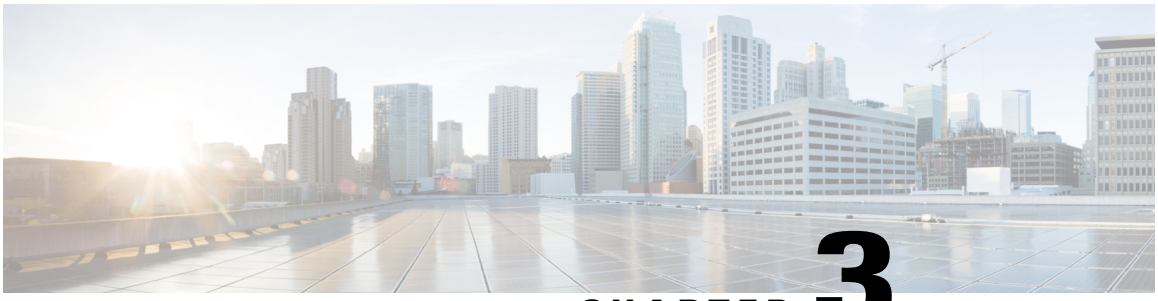
Step 4 From the topology, select the source device, adjacent devices, and then the destination device.

Step 5 In the LSP Name field, enter a name for the new LSP and then click **Deploy**.

Deleting LSPs

Step 1 From the upper right-hand corner of the content pane, click the Establish LSP icon.
A listing of the LSPs that are currently active is displayed.

Step 2 Select the LSP you want to delete and then click **Delete LSP**.



CHAPTER 3

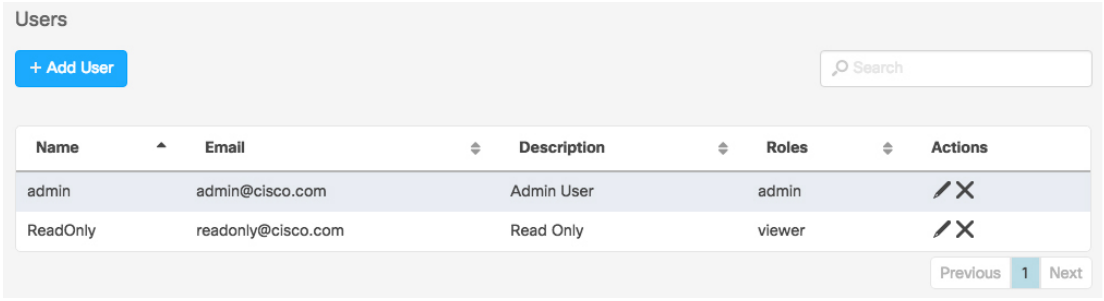
Managing Your System

Open SDN Controller allows you to manage both the features that are available to the controller (via the Features Management page) and the users that will make use of those features (via the User Management page). The following topics describe how to do so:

Managing Users

From the Users page, you can add new Open SDN Controller users, edit the settings for existing users, and delete users from the system. To open this page, select **Users** from the main toolbar's Management menu.

Figure 13: Users Page



Adding a User

Step 1 From the Users page, click **Add User** to open the Add a New User dialog box.

Step 2 Specify the following information for the new user:

- Username-Note that you will not be able to update this setting once the user has been added.
- Description (optional)
- Email address
- Passphrase
- Role-Specify whether the user will have administrator privileges.

Step 3 Click **Save** to save the settings for the new user.
The user should now be listed on the Users page.

Editing a User

Step 1 From the Actions column in the Users page, click the Edit icon for the user whose settings you want to edit.
The Edit User dialog box opens.

Step 2 Make the necessary changes to any of the following user settings:

- Description
- Email address
- Passphrase
- Role

Note that you cannot change a user's username.

Step 3 Click **Save** to save your changes.

Deleting a User

Step 1 From the Actions column in the Users page, click the Delete icon for the user you want to delete from the system.

Step 2 Click **Delete** to confirm the deletion of that user.

Managing Features

From the Features page, you can view a listing of every Open SDN Controller feature that is currently installed on the system. You can also determine which of these features are active, install new features, update existing features, and activate features as needed. To open the Features page, select **Features** from the main toolbar's Management menu.

Identifying Active Features

To determine whether a feature is currently active, simply locate its entry in the Features table and look at the Active column. If the feature is active, a check mark is displayed here.

Installing New Features

Step 1 From the Features page, click **Manage Features**.
The Manage Features dialog box opens.

Step 2 Click **Add Features**.

Step 3 Click **Browse...** and navigate to the feature's installer (.kar) file.

Step 4 Select the file and then click **Open**.

Step 5 Specify the type of feature you are installing:

- If the feature updates the controller's GUI, select the User Interface radio button.
- If the feature updates the controller's backend, select the Controller radio button.

Step 6 Click **Save**.

Step 7 Verify that the feature you just installed is listed in the Features table.
Note the following:

- It may take a minute or two for the feature you installed to be listed.
- Once a feature has been installed, it cannot be removed.
- When you first install a feature, you can continue without restarting the controller. Open SDN Controller only restarts after you update an existing feature.

Updating Existing Features

Step 1 From the Features page, click **Manage Features**.

The Manage Features dialog box opens.

Step 2 Locate the feature you want to update and then click its Update icon in the Actions column.

Step 3 Click **Browse...** and navigate to the feature's installer (.kar) file.

Step 4 Select the file and then click **Open**.

Note You will not be able to change the file's feature type.

Step 5 Click **Save**.

Note the following:

- When you update a feature, Open SDN Controller automatically restarts and may take up to 20 minutes to become operational again.
- After the restart, you will need to activate any features you manually activated previously.

Activating Features

Step 1 In the Features table, locate the entry for the feature you want to activate.

Step 2 In the Actions column, click the Activate icon and then verify that a check mark is displayed in the Active column.

Note Open SDN Controller will indicate when an error has occurred while activating a feature.

Managing Open SDN Controller Licenses

From the License Information page, you can view a listing of every license that is installed on the controller, view the settings configured for those licenses, and specify the settings for new licenses. Open SDN Controller provides the following information for each license:

- Key
- Version
- Description
- Number of supported users

Adding a License

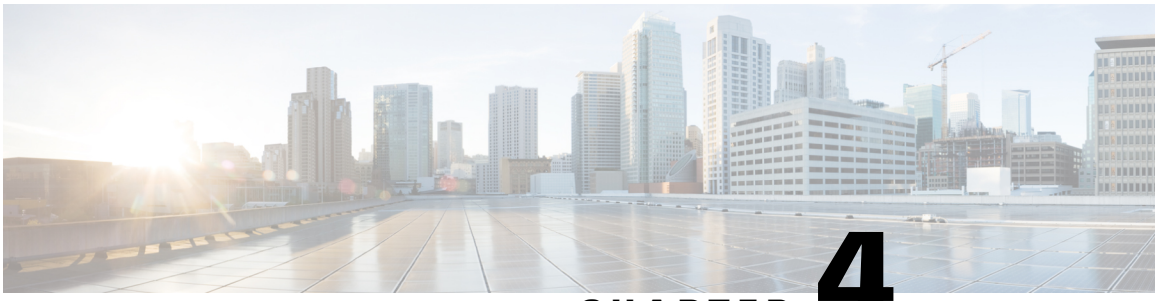
Step 1 From the main toolbar's Management menu, select **Licenses**.

Step 2 Enter the following information:

- License server URL
- License token ID
- Software ID tag
- Transport mode
- Product ID

Step 3 Click **Save**.

Step 4 Verify that the license you just added is listed in the table at the bottom of the License Information page.



CHAPTER 4

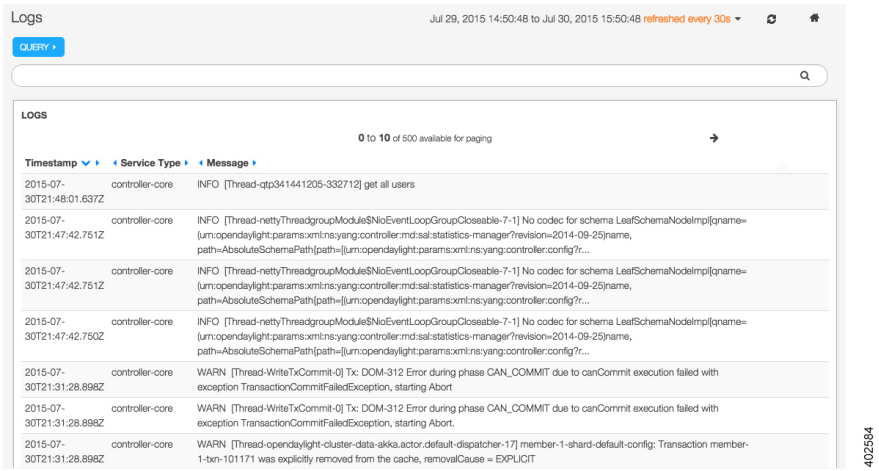
Monitoring Your System

Open SDN Controller provides three pages that allow you to monitor the health and performance of your system: the Logs Dashboard page, the Metrics Dashboard page, and the Services Status page. The following topics describe these pages in more detail:

Viewing the Logs Dashboard

From the Logs Dashboard, you can view information for the events that have taken place in your system. To open the Logs Dashboard, select **Logs** from the main toolbar's Monitoring menu.

Figure 14: Logs Dashboard



Logs Dashboard Components

The following table describes the components that make up the Logs Dashboard.

Component	Description
Toolbar	From here, you can: • Set the timeframe for which information is displayed in the dashboard • Set how often the dashboard's information is automatically refreshed • Manually refresh the dashboard's information • Revert to the default dashboard layout by clicking the Go to saved default (house) icon
Query field	Allows you to search for event information that contains a particular string. See Running Queries for more information.
Logs widget	Lists the 500 latest events that have taken place in your system. See Viewing Log Events for more information.
Log Summary widget	Indicates the number of events (grouped by severity) that have taken place over the timeframe currently set for the Logs Dashboard. To determine the number of events that are of a specific severity, place your cursor over the corresponding bar in the graph.
Component Summary widget	Indicates the component or device from which events originated and the total number of events that took place on that component or device.
Log Activity widget	Visualizes the number of events that have occurred over the timeframe currently set for the Logs Dashboard. To determine the exact number of events that took place at a certain time, place your cursor over the corresponding bar in the timeline.

Running Queries

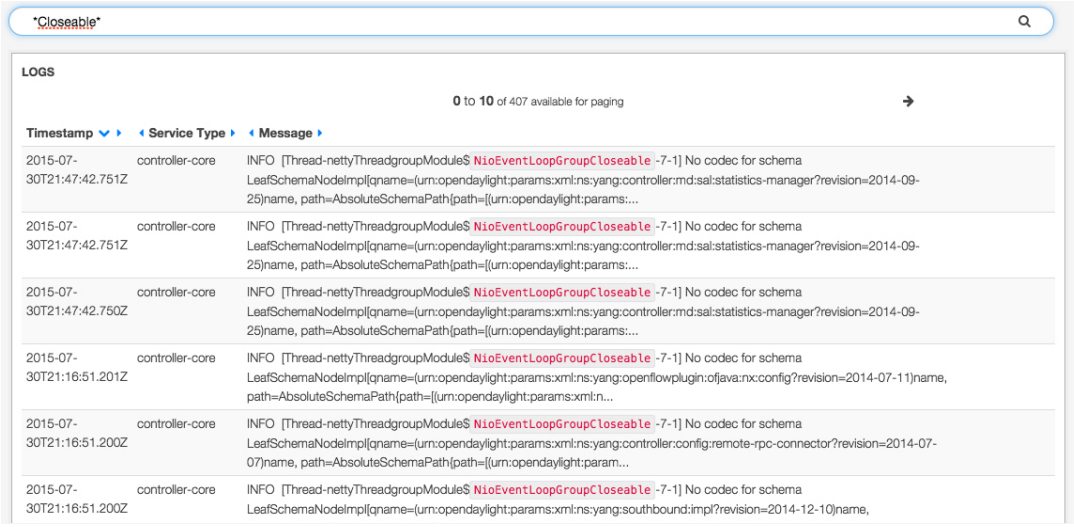
By specifying a query, you can view only the event information that contains a particular string. To run a query, enter the appropriate text in the Query field and then click the Search icon or press the Enter key.

Note the following:

- As you type the string you want to search for, Open SDN Controller suggests additional strings that you can select and search for instead.

- To search for a string that is part of a longer string, enclose it within asterisks. For example, entering ***Closeable*** returned the results displayed in the following screenshot. If you had entered **Closeable** instead, only event information that contained *Entries* as a separate word would have been returned.

Figure 15: Sample Query



402585

- To clear the results of a query you have run, empty the Query field and then click the Search icon or press the Enter key.
- You can toggle the Query field on and off by clicking the Query button.

Creating Filters

You can create a filter in two Logs Dashboard components: the Logs widget and the Component Summary widget.

From the Logs Widget

Step 1 Click the table entry for the event you want to base a filter on.
The Logs table updates, displaying all of the fields available for that event.

Step 2 Locate the field that contains the value you want to base a filter on.
For example, say you want to view only events with a severity of 4. In this case, you would need to locate the @fields.Severity table entry.

Step 3 In the field's Action column, click the Add filter to match this value icon.
Note that you can create and apply multiple filters to the information displayed in the Logs Dashboard.

From the Component Summary Widget

Step 1 Locate the event source you want to base a filter on.

Step 2 In the Filter By column, click the filter icon to view only the events that originated from that source.

Setting the Logs Dashboard Timeframe

Do one of the following to change the timeframe for which information is displayed in the Logs Dashboard:

- At the top of the dashboard, click the link for the timeframe that is currently displayed in the dashboard. In the resulting drop-down list, select the desired timeframe. If you want to specify a timeframe that is not covered by one of the available options, select **Custom**, specify the desired timeframe, and then click **Apply**.

  Note From this drop-down list, you can also select **Auto-Refresh** and specify how often the information displayed by these graphs is automatically refreshed. To manually refresh this information, click the Refresh icon.

- In the Log Activity widget, click the desired start time. While holding down the mouse, drag the cursor to the desired end time and then release the mouse.

Viewing Log Events

From the Logs widget, you can view a listing of the 500 most recent events that have taken place in your system.

To set which fields are displayed here:

Step 1 Click the table entry for any event listed in the Logs table.
The table updates, displaying all of the fields that are available and their current values.

Step 2 Locate the table entry for the field you want the Logs table to display.

Step 3 From the Action column, click the Toggle table column icon.
Repeat these steps to remove a field from the Logs table.

Viewing Controller Metrics

From the Metrics Dashboard, you can view graphs that visualize the following performance metrics for the controller, helping you to identify any issues that require attention:

- CPU usage
- Memory usage
- CPU load
- Heap size
- Network usage
- Free disk space

Figure 16: Metrics Dashboard

If multiple controller nodes are set up in your system, a separate graph for each of these metrics is displayed for each node.

To open the Metrics Dashboard, select **Metrics** from the main toolbar's Monitoring menu.

Do one of the following to change the timeframe for which information is displayed in the graphs:

- From the top of the Metrics Dashboard, click **Zoom Out**. Every time you click this link, the timeframe these graphs cover is expanded.
- To the right of the Zoom Out link, click the link for the timeframe that is currently displayed in the graphs. From the resulting drop-down list, select the desired timeframe. If you want to specify a timeframe that is not covered by one of the available options, select **Custom**, specify the desired timeframe, and then click **Apply**.

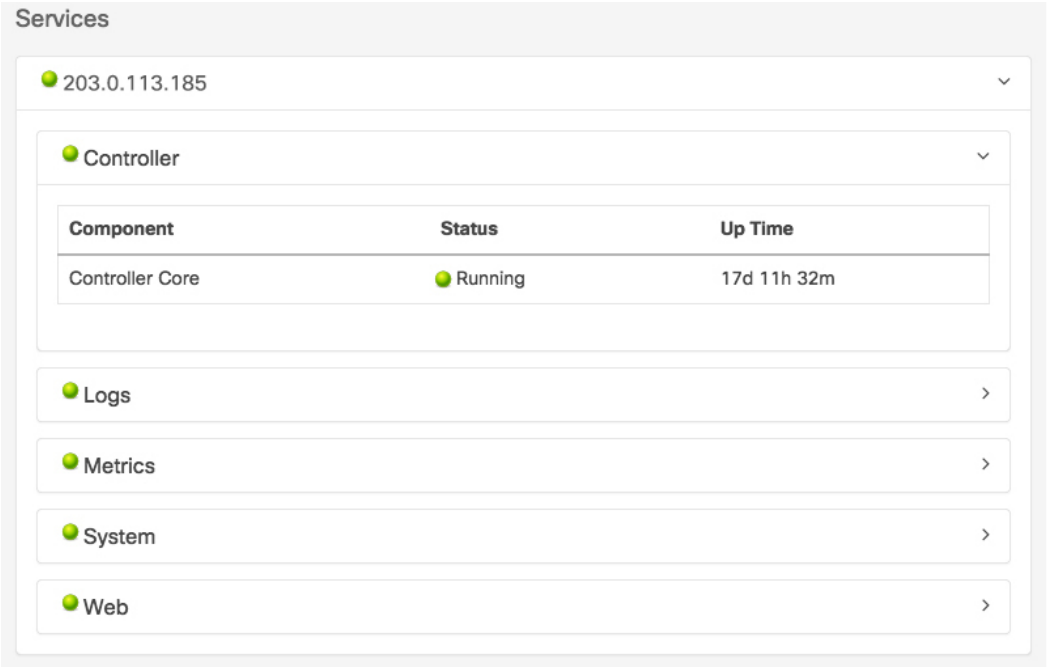
Note From this drop-down list, you can also select **Auto-Refresh** and specify how often the information displayed by these graphs is automatically refreshed. To manually refresh this information, click the Refresh icon.

- In any of the graphs, click the desired start time. While holding down the mouse, drag the cursor to the desired end time and then release the mouse.

Viewing Services Status

From the Services page, you can view the services installed on a controller node, determine whether they are running and, for the services that are running, see how long they have been up. By default, this page is open after you log into Open SDN Controller. To open the Services page when another page is open, select **Services** from the main toolbar's Monitoring menu.

Figure 17: Services Page



Note By default, the information displayed on this page is automatically refreshed every 10 seconds.

Complete the following procedure to determine whether any services are down and need to be restarted.

1 View the ball icon that precedes a controller node's IP address.

- If the icon is green, this indicates that all of the services on the node running. You can stop here.

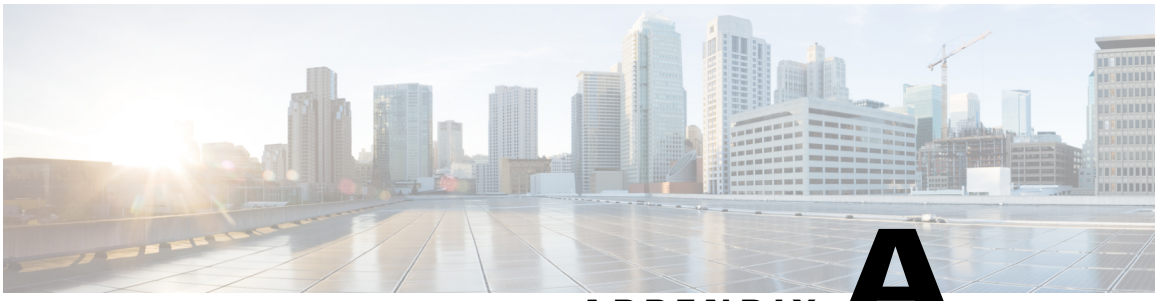
- If the icon is yellow, this indicates that one or more services are down on the node. Proceed to Step 2.

2. If necessary, click the node's link to bring up a listing of the four components for which service status is tracked:
   - Metrics framework
   - Logs framework
   - System
   - Controller
3. Click any component that is preceded by a red ball icon to view a listing of the services installed on that component.
4. Restart any services that are currently down (indicated by a red ball icon).

Exporting Diagnostic Information

Step 1 From the main toolbar's Help menu, select **Export Diagnostic Data**.
The zipped TAR file (diagnostic-data.tgz) is downloaded to your default download directory.

Step 2 Unzip the TAR file to the desired directory.
The latest diagnostic information for your system is now available for you to view offline.



APPENDIX A

RESTCONF Requests

RESTCONF is a REST-like protocol that provides access over HTTP to the two data stores that the controller maintains: the config data store and the operational data store. While you can use the Open SDN Controller GUI to add, modify, delete, or retrieve information stored within those data stores, there are some items that you can access or perform only by submitting a RESTCONF request. We recommend you use a REST client (such as Postman) to simplify the submission of RESTCONF requests.

The following topics describe how to make a RESTCONF request and provide examples of requests that you may need to make:

Making RESTCONF Requests

To make a request using Postman:

Step 1 Open Postman.

Step 2 Generate a security token:

a) In the Enter request URL here field, enter the following URL:
https://<*controller-IP-address*>/controller-auth?grant_type=password&username=admin&password=<*admin-user-password*>&scope=sdn

b) From the methods drop-down list, select **POST**.

c) Click **x-www-form-urlencoded**.

d) Click **Send**.

e) In the Body tab, locate and copy the access_token value (do not include the quotation marks).

Step 3 Select the Basic Auth tab and then do the following:

a) In the Username field, enter the admin user's username.

b) In the Password field, paste the access_token value you copied in the previous step.

Step 4 Replace the URL you previously entered with one that is structured as follows:
https://token:$token@<*controller-IP-address*>/controller/URI

where *URI* is the appropriate uniform resource identifier.

This is where you will enter the URL specified for requests in Open SDN Controller's administrator and installation guides.

Step 5 Select the appropriate HTTP method from the methods drop-down list.

Step 6 Click **raw** and then select **XML** from the data format drop-down list.
Note If you are using Postman 2.0, select **XML (application/xml)** instead.

Step 7 Enter the text for the request and then click **Send**.

Sample RESTCONF Requests

The following topics provide RESTCONF requests that you may need to make during the day-to-day administration of your network.

Adding BGP Devices

To add devices to the BGPLS Manager topology, complete the following procedure. Before you continue, note the following:

- You need to complete Step 4 for every BGP application peer that is connected to a BGP speaker.
- In Step 4b, the value you specify for the bgp-peer-id parameter is used by the BGP Best Path Selection algorithm.
- Before every RESTCONF request you make, you must first generate a security token. See Making RESTCONF Requests for more information.

Step 1 Configure the RIB module on a BGP speaker by making the following POST request, updating the values for the bgp-rib-id and local-as parameters with the correct values for the BGP speaker:

(URL—use this for all of the requests in this procedure)

https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/controller-config/

yang-ext:mount/config:modules/

(Payload)

```
<module xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:rib-impl</type>
  <name>example-bgp-rib</name>
  <bgp-rib-id
xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">192.0.2.2</bgp-rib-id>
  <local-as xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">64496</local-as>
</module>
```

Step 2 If the BGP speaker supports linkstate attribute type 29, make the following POST request to change the iana-linkstate-attribute-type value to *true*. Otherwise, proceed to Step 3.

(Payload)

```
<module xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
 <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:linkstate">x:bgp-linkstate</type>

 <name>bgp-linkstate</name>
 <iana-linkstate-attribute-type
xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:linkstate">true</iana-linkstate-attribute-type>
</module>
```

Step 3 Configure the bgp-peer module on the BGP speaker by making the following POST request, updating the values for the host and holdtimer parameters for the module (if necessary):

(Payload)

```
<module xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
 <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:bgp-peer</type>
 <name>example-bgp-peer</name>
 <host xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">192.0.2.1</host>
 <holdtimer xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">180</holdtimer>
 <rib xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:cfg">x:rib</type>
  <name>example-bgp-rib</name>
 </rib>
 <peer-registry xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type
xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:bgp-peer-registry</type>
  <name>global-bgp-peer-registry</name>
 </peer-registry>
 <advertized-table xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:bgp-table-type</type>

  <name>ipv4-unicast</name>
 </advertized-table>
<advertized-table xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:bgp-table-type</type>

  <name>ipv6-unicast</name>
 </advertized-table>
 <advertized-table xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:bgp-table-type</type>

  <name>linkstate</name>
 </advertized-table>
</module>
```

Step 4 Register every BGP application peer that is connected to the BGP speaker you just configured:

a) Configure the RIB module on a BGP application peer by making the following POST request, updating the values for the bgp-rib-id and local-as parameters with the correct values for the BGP application peer:

(Payload)

```
<module xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:rib-impl</type>
  <name>example-bgp-rib</name>
```

```
  <bgp-rib-id
xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">192.0.2.2</bgp-rib-id>
  <local-as xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">64496</local-as>
</module>
```

b) Configure the BGP application peer by making the following POST request, updating the values in bold with the correct values for the BGP application peer:

(Payload)

```
<module xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
 <type
xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:bgp-application-peer</type>

 <name>example-bgp-peer-app</name>
 <bgp-peer-id
xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">203.0.113.5</bgp-peer-id>
 <target-rib xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">x:rib-instance</type>

  <name>example-bgp-rib</name>
  </target-rib>
 <application-rib-id
xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">example-app-rib</application-rib-id>

 <data-broker xmlns="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">
  <type
xmlns:x="urn:opendaylight:params:xml:ns:yang:controller:md:sal:binding">x:binding-async-data-broker</type>

  <name>pingpong-binding-data-broker</name>
 </data-broker>
</module>
```

OpenFlow RESTCONF Requests

The following RESTCONF requests are the ones you will most likely make for the OpenFlow-enabled devices in your network. Before every request you make, you must first generate a security token. See Making RESTCONF Requests for more information.

Retrieving the Controller's Inventory Database

HTTP method—GET

URL—https://token:$token@<*controller-IP-address*>/controller/restconf/operational/opendaylight-inventory:nodes

Retrieving the Controller's Topology Database

HTTP method—GET

URL—https://token:$token@<*controller-IP-address*>/controller/restconf/operational/network-topology:network-topology/

Adding a Flow to a Device

HTTP method—PUT

URL—https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/openflow:<*OpenFlow-device-ID*>/table/<*table-ID*>/flow/<*Flow-ID*>

Payload—

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <flow-name>Flow1</flow-name>
    <id>1</id>
    <cookie_mask>255</cookie_mask>
    <cookie>1</cookie>
    <idle-timeout>1000</idle-timeout>
    <table_id>0</table_id>
    <priority>2</priority>
    <hard-timeout>1200</hard-timeout>
    <installHw>false</installHw>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <output-action>
                        <output-node-connector>49</output-node-connector>
                        <max-length>60</max-length>
                    </output-action>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <match>
        <ethernet-match>
            <ethernet-type>
                <type>2048</type>
            </ethernet-type>
            <ethernet-destination>
                <address>ff:ff:ff:11:12:13</address>
            </ethernet-destination>
            <ethernet-source>
                <address>aa:aa:aa:11:12:13</address>
            </ethernet-source>
        </ethernet-match>
        <ipv4-source>192.0.2.211/8</ipv4-source>
        <ipv4-destination>203.0.113.137/16</ipv4-destination>
        <ip-match>
            <ip-protocol>6</ip-protocol>
            <ip-dscp>2</ip-dscp>
        </ip-match>
        <tcp-source-port>25364</tcp-source-port>
        <tcp-destination-port>8080</tcp-destination-port>
    </match>
</flow>
```

Note the following:

- The values provided here are sample values. Please enter the appropriate values for your setup.
- In the request URL, ensure that the values you specify for the flow table ID and flow ID match the values specified for these settings in the request's payload.

Retrieving a Flow from the Configuration Datastore

HTTP method—GET

URL—https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/openflow:

<*Openflow-device-ID*>/table/<*table-ID*>/flow/<*Flow-ID*>

Deleting a Configuration Datastore Flow

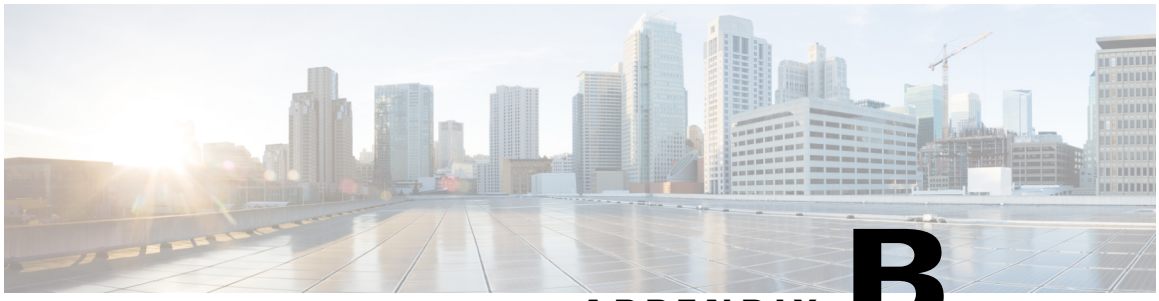
HTTP method—DELETE

URL—https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/openflow:<*OpenFlow-Device-ID*>/table/<*table-ID*>/flow/<*flow-ID*>

Retrieving a Flow from the Operational Datastore

HTTP method—GET

URL—https://token:$token@<*controller-IP-address*>/controller/restconf/operational/opendaylight-inventory:nodes/node/openflow:<*OpenFlow-device-ID*>/table/<*table-ID*>/flow/<*flow-ID*>



APPENDIX B

OpenFlow Clusters

In Open SDN Controller, OpenFlow clusters work with datastore clusters to provide high availability (HA). So what is a cluster? In this context, a cluster is a collection of datastores that work together as if they are a single entity, even though they reside on different controller nodes.

Refer to the following topics for an overview of both OpenFlow clusters and the flow modification process, as well as troubleshooting information:

Cluster Management Overview

Open SDN Controller supports two versions of OpenFlow: versions 1.3 and 1.0. Let's briefly cover the similarities and differences in how each version manages clusters.

OpenFlow 1.3

In OpenFlow 1.3, each switch is connected to every controller node that belongs to a cluster. The switch assigns one of the following roles to each controller node:

- **Master—**All synchronous and asynchronous messages are sent to the master controller node. This node has write privileges on the switch.
- **Slave—**Only synchronous messages are sent to this controller node. Slave nodes only have read privileges on the switch.
- **Equal—**When this role is assigned to a controller node, that node has the same privileges as the master node. By default, controller nodes are assigned the Equal role when they first connect to the switch.

Each datastore on the controller nodes is divided into smaller chunks known as shards, and one of these shards will act as the leader. For example, the inventory-operational-shard is present in the inventory datastore for all of a cluster's nodes. One of these shards will act as the leader, with the other shards operating as followers. This is important because the node on which the inventory-operational-shard leader resides is assigned as the master node to the switch connected to the cluster.

OpenFlow 1.0

Since OpenFlow 1.0 does not support roles, the switch that is connected to a cluster is only connected to one controller node at any given time (via the use of a floating/virtual IP address). In the event that the node connected to the switch goes down, the switch is automatically connected to another node which is then elected as the inventory-operational-shard leader.

Just like in OpenFlow 1.3, each datastore on the controller nodes is divided into shards, with one of these shards acting as the leader. This is important because the floating/virtual IP address for this cluster is configured to point to the node on which the inventory-operational shard leader resides (the master node).

Flow Modification Process

There are two types of in-memory datastores: config and inventory. And on each of these datastores, four shards reside: default, inventory, toaster, and topology. Of note for flow modifications are the inventory shards in both the config and operational datastore. In the following topic, we'll cover what happens when the inventory-config-shard leader and inventory-operational-shard leader reside on separate cluster nodes and a flow is modified.

1. Flows are first added to the inventory-config-shard leader.

   The addFlow request is routed from the controller node that made the request to the node on which the inventory-config-shard leader resides (the master node).

2. When the flow is received by the inventory-config-shard leader, the leader replicates the flow and then commits it to the datastore.

3. A notification is generated when the flow is submitted to the datastore and an addFlow remote procedure call (RPC) is sent to the remote RPC connector.

   All switch RPCs are registered only on the master node.

4. The Remote RPC connector locates the master node and forwards the addFlow RPC to it.

5. When the RPC component of the master node receives the addFlow RPC, it forwards the RPC to the OpenFlow plug-in, which in turn forwards the RPC to the switch.

6. In the background, the Statistics Manager regularly polls the switch by executing flow and other statistics RPCs against the switch. The Statistics Manager is enabled only on the master node.

7. The switch responds to these RPCs with notifications.

   These notifications are sent only to the master node.

8. When these notifications are received, the Statistics Manager adds the flows to the inventory-operational datastore.

OpenFlow Clusters FAQs and Troubleshooting

How do I determine the role of each controller node known to the switch?

- For OVS switches, run the following command:

  sudo ovs-vsctl list CONTROLLER

- For other types of switches, refer to the switch's documentation for the appropriate command.

The wrong role is assigned to a controller node.

1 Check the status of the last role change in the OF Role Service by running the broadcastRoleChange script. To do so, make a POST request, using the following URL:

https://token:$token@<*controller-IP-address*>/controller/restconf/operations/of-operational-status:get-operational-status

Note the following:

- Before every RESTCONF request you make, you must first generate a security token. See Making RESTCONF Requests for more information.
- If the request returns `RUN`, this indicates that the controller has submitted a role change request to the switch in order to assume the Master role.
- If the request returns `STANDBY`, this indicates that the controller has submitted a role change request to the switch in order to assume the Slave role.
- You need to make this request 3 times: once for each controller node in the cluster. The status for one node should be `RUN` and `STANDBY` for the other two. If this is not the case, check the log file for any related errors.

2 Navigate to the /opt/cisco/controller/bin/role_scripts directory and open the broadcastRoleChange.sh.log file:

- If the last line of this file lists `RUN`, this indicates that the Openflowplugin Orchestration app received the role change request, executed the broadcastRoleChange script, and made a call to the OF Role Service.
- If the broadcastRoleChange.sh.log file is not present, this indicates that the Openflowplugin Orchestration app never received the role change request. Open the log file and look for any related errors.

How do I determine a controller node's role without accessing the switch?

Do one of the following:

- In JConsole, identify the controller node on which the inventory-operational-shard leader resides. This is the Master controller node.
- Open the broadcastRoleChange.sh.log file (located in the /opt/cisco/controller/bin/role_scripts directory) and check the last line:
  - If `RUN` is listed, the controller is the Master node.
  - If `STANDBY` is listed, the controller is a Slave node.

I don't see certain flows programmed on the switch in either the inventory-operational database or OpenFlow Manager.

1 Check whether the flow was programmed to the switch.

2 Open Jconsole and connect to the Master controller node (on which the inventory-operational-shard leader resides).

3 Select **org.opendaylight.controller** > **StatisticsManager** > **Switches**.

The ID/name of the switch is displayed.

4 Select **ID** > **Attributes** and determine when the LastFlowStatsPollTime and LastFlowStatsNotificationReceived objects were last accessed.

5 Wait 15–20 seconds and then refresh the Attributes screen.

If the attributes remain unchanged, this indicates that the flow's statistics are not being updated. Check the logs for any related errors.

I have added a flow but it does not appear on the switch.

Before you complete the following procedure, we recommend that you review the "How do flow mods take place in the clustered environment" topic.

1 Check whether the flow is available in the switch's inventory-config datastore by opening the following URL:

http://<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes

where <*controller-IP-address*> is the IP address for any controller in the cluster.

- If it is, check the logs for any related errors.
- If it isn't, proceed to Step 2. Make sure to note the switch's ID, which you can find in the following line in the XML payload:

```
<node><id><switch-ID></id></node>
```

You will need this for Step 4.

2 Find the node on which the inventory-config-shard leader resides:

a Open JConsole and connect to a controller node.

b Select **org.opendaylight.controller** > **DistributedConfigLeader** > **member-1-shard-inventory-config** > **Attributes** and look for the Leader attribute.

c Connect to the node tagged with the Leader attribute (unless you are already connected to it).

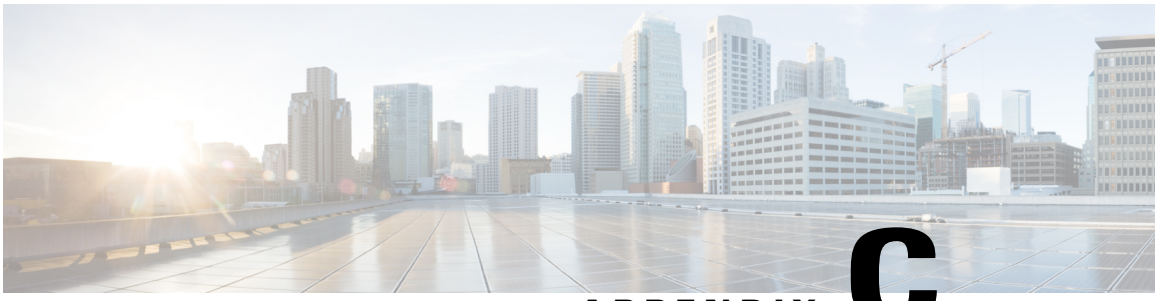
3 Select **org.opendaylight.controller** > **RemoteRpcBroker** > **Operations** to open the RemoteRpcBroker screen.

4 In the findRpcByRoute section, enter the switch's ID.

A popup window opens.

5 Look for the addFlow RPC:

- If you cannot find it, look for RPC registration errors in the logs.
- If you do find it, note its value:
  - If the value is `local`, this indicates that the inventory-operational-shard leader and inventory-config-shard leader reside on the same node. Verify this by selecting **org.opendaylight.controller** > **DistributedConfigLeader** > **member-1-shard-inventory-operational** and then checking the logs for any addFlow errors that have occurred on this node.
  - If the value is an IP address or hostname, this indicates that the inventory-operational-shard leader resides on that device. Connect to that device via Jconsole and go back to Step 3 of this procedure.



APPENDIX C

Open SDN Controller Security

The following topics describe the security measures that Open SDN Controller implements:

Security Considerations

There are three levels of security built into Open SDN Controller: OS-level security, application-level security, and API-level security. This topic covers the security measures that are in place for each of these levels and describes any potential vulnerabilities that you should be aware of.

OS-Level Security

At the OS level, there are two main attack vectors: VM console access and SSH access. Console access is subject to VMware security measures and assumes that the client is following the guidelines VMware recommends to secure your VM console. SSH access is protected because root logins are not allowed and SSH access is disabled for all users except the sysadmin user (a user with less privileges). In addition, Open SDN Controller forces the sysadmin user to change their password after logging in for the first time and enforces password complexity requirements.

The main security vulnerability at the OS level is that the sysadmin user has sudo privileges. As a result, if the password is ever compromised, that user can get sudo root access to the system.

Application-Level Security

To address the application attack vector, Open SDN Controller redirects all HTTP traffic from port 80 to port 443, which is configured to use HTTPS to handle data. The controller also uses HTTPS to encrypt all passwords.

The main security vulnerability at the application level is that user passwords are stored in Open SDN Controller's database, meaning that the controller and user passwords reside in the same location.

API-Level Security

At the API level, Open SDN Controller uses HTTPS to handle HTTP traffic. It also minimizes password exposure in API calls by generating a token hash of the password for every call that is made. As a result, REST API calls and the password are not stored together.

Configuring LDAP

Open SDN Controller supports the use of your company's Lightweight Directory Access Protocol (LDAP) server for authentication. To enable this functionality, complete the following procedure.

Step 1 Run the following commands to shut down the monit and controller services:

- **sudo service monit stop**
- **sudo service controller stop**

Step 2 Navigate to the following directory: /opt/cisco/controller/etc

Step 3 In a text editor, open the LDAP server configuration file (ldap.cfg).

Step 4 Locate the following settings and set the values that are specified:

- **ldap-timeout:** 3000
- **ldap-enable:** true
- **ldap-dn:** <*company-distinguished-name*>
- **ldap-ssl-port:** <*SSL-port-number*>
- **ldap-nossl-port:** <*noSSL-port-number*>
- **ldap-use-ssl:** true
- **ldap-object-group:** <*company-object-group*>
- **ldap-host:** <*LDAP-server-hostname*>

If necessary, consult your company's IT department to determine the correct values for the ldap-dn, ldap-ssl-port, ldap-nossl-port, ldap-object-group, and ldap-host settings.

Step 5 Save the changes you have made and then restart the controller.
You should now be able to log into Open SDN Controller with the username and password you use to access your company's network.

Configuring a RADIUS Server for AAA Authentication

In OSC 1.1, you can configure a RADIUS server to implement AAA authentication. There are a number of commercial and open source RADIUS servers available for you to choose from. The following topics assume that you are configuring the FreeRadius server.

Adding a New RADIUS Server Client

The RADIUS protocol is based on UDP. Since UDP does not make use of connections, it cannot use SSL or another type of encryption based on TCP connections to handle communications. To work around this, each client that wants to use the RADIUS server for authentication must be predefined and added to the server. In FreeRadius, you accomplish this by updating the clinet.cfg file, which is located in the /etc/freeradius directory.

Note If you are using the RadiusDesk suite, the directory in which clinet.cfg resides will differ.

To add a new client, locate the following parameters in the RADIUS server's client.cfg file and define values for them:

- *client*—client's hostname
- *ipaddr*—client's IP address
- *secret*—password-like value assigned to the client

Here is what a sample client configuration looks like. The values you need to specify are italicized:

```
client cosc-ova-181 {
  ipaddr = 192.0.2.122
  secret = cosc
}
```

Configuring OSC to Use a RADIUS Server

The RADIUS configuration file, radius.cfg, is located in the /opt/cisco/controller/etc directory. It is an active file, which means that any changes made to it will automatically be rolled into OSC at runtime. As a result, you do not need to restart the controller after you edit the configuration file.

Here is an example of what the configuration file looks like:

```
radius-secret=cosc
radius-enable=true
radius-host=198.51.100.137
```

where *radius-secret* indicates the secret you defined for this client, *radius-enable* indicates whether RADIUS integration has been enabled, and *radius-host* indicates the RADIUS server's IP address.

After you have enabled RADIUS, you will be able to log into OSC with any defined RADIUS username and password combination. Note that a local OSC user is created from the RADIUS user and is assigned the User role. If you want to change the RADIUS user's role to *Admin*, you need to log into OSC as an admin user and then change that user's role from the Users page.

Setting Up TLS Support

Complete the following procedure to set up TLS support on either a Nexus 3000 Series or Catalyst 4000 Series switch.

Step 1 Complete basic setup tasks.

a) In a directory on the controller's VM, create a subdirectory named *tls*:

- **cd** <*controller-VM-directory*>
- **bash**
- **mkdir tls**
- **cd tls**

b) Create directories for the Certification Authority (CA) certificates, private key, and CRL:

- **mkdir -p mypersonalca/certs**
- **mkdir -p mypersonalca/private**
- **mkdir -p mypersonalca/crl**
- **mkdir -p controller**
- **mkdir -p of-switch**

c) Initialize the CA database:

- **echo "01" > mypersonalca/serial**
- **touch mypersonalca/index.txt**

d) In the TLS root directory, create a file named ca.cnf (the OpenSSL configuration file) and ensure it contains the following information:

```
[ ca ]
default_ca = mypersonalca

[ mypersonalca ]
#
# WARNING: If you modify this parameter, ensure that you specify the same directory for the
default_keyfile parameter (in the [req] section below).
# where everything resides
dir = ./mypersonalca

# where issued certificates reside
certs = $dir/certs

# where issued CRLs reside
crl_dir = $dir/crl

# database index file
```

```
database = $dir/index.txt

# default directory for new certificates
new_certs_dir = $dir/certs

#
# CA certificate
certificate = $dir/certs/ca.pem


# current serial number
serial = $dir/serial

# current CRL
crl = $dir/crl/crl.pem

# WARNING: If you modify this parameter, ensure that you specify the same directory for the
default_keyfile parameter (in the [req] section below).
# private key
private_key = $dir/private/ca.key

# private random number file
RANDFILE = $dir/private/.rand

# extensions to add to the certificate
x509_extensions = usr_cert

# how long to certify the certificate for
default_days = 365

# how long before the next CRL
default_crl_days= 30

# which MD to use
default_md = sha1

# keep passed DN ordering
preserve = no

# section names
policy = mypolicy
x509_extensions = certificate_extensions

[ mypolicy ]
# We recommend that you do not change these values.
commonName = supplied
stateOrProvinceName = optional
countryName = optional
emailAddress = optional
organizationName = optional
organizationalUnitName = optional

[ certificate_extensions ]
# The signed certificate cannot be used as the CA.
```

```
basicConstraints = CA:false

[ req ]
# same as the private_key
default_keyfile = ./mypersonalca/private/ca.key

# specify which hash to use
default_md = sha1

# enable/disable prompts
prompt = no


# This is for CA.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer
string_mask = utf8only
basicConstraints = CA:true
distinguished_name = root_ca_distinguished_name
x509_extensions = root_ca_extensions

[ root_ca_distinguished_name ]
# update with the appropriate values for your organization.
commonName = Controller
stateOrProvinceName = Mass
countryName = US
emailAddress = root_ca_userid@cisco.com
organizationName = Cisco

[ root_ca_extensions ]
basicConstraints = CA:true
```

e) Create additional directories for the CA certificates, private key, and CRL:

- **cp ca.cnf ca_main.cnf** (ca_main.cnf acts as a backup file for ca.cnf)
- **sed s/root_ca_userid/`whoami`/ <./ca_main.cnf >./ca.cnf**
- **setenv OPENSSL ca.cnf** (for tcsh)
- **export OPENSSL="ca.cnf"** (for bash)

f) (Optional) Clean up the directories you have created before creating a new certificate in the TLS workspace:

- **cd tls**
- **rm -rf mypersonalca/index***
- **rm -rf mypersonalca/serial***
- **rm -rf mypersonalca/certs/***
- **rm -rf mypersonalca/private/***
- **rm -rf sw-cert.pem**
- **rm -rf of-switch/***

- **rm -rf controller/***

Step 2 Create the CA certificate (ca.pem) and private key (ca.key):

a) Run the following commands:

- **cd tls**
- **openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out ./mypersonalca/certs/ca.pem -outform PEM -keyout ./mypersonalca/private/ca.key**

b) When prompted, enter the required information (such as your organization's name and your email address).

Step 3 Copy the CA certificate to the of-switch directory:
cp ./mypersonalca/certs/ca.pem ./of-switch/sw-cacert.pem

Step 4 Create the CA certificate and private key for the controller:

a) Create the controller's private key (ctl-privkey.pem) and certificate request (ctl-cert.req):

1. Run the following command:

   openssl req -nodes -newkey rsa:2048 -keyout ./controller/ctl-privkey.pem -keyform PEM -out ./controller/ctl-cert.req -outform PEM

2. When prompted, enter the required information (such as your organization's name and your email address).

b) Create the controller's CA certificate (ctl-cert.pem):
openssl ca -batch -notext -in ./controller/ctl-cert.req -out ./controller/ctl-cert.pem -config ./ca.cnf

Step 5 Verify that the certificate is valid:

a) From the controller, determine the certificate's start date and time:
openssl x509 -in ./controller/ctl-cert.pem -text | grep Not

b) From a Nexus 3000 Series switch, determine the certificate's start date and time:
sh clock

The certificate is valid when the start date and time indicated on the controller precedes the date and time indicated on the Nexus 3000 Series switch.

Step 6 Configure TLS support on your device.

- For Nexus 3000 Series switches, complete the procedure described Configuring TLS Support on a Nexus 3000 Series Switch.
- For Catalyst 4000 Series switches, complete the procedure described Configuring TLS Support on a Catalyst 4000 Series Switch.

Step 7 Configure TLS support in OSC's Openflow configuration file.
In this example, we will assume that your controller's root directory is /opt/cisco/controller/.

a) Copy ctl-cert.pem, ctl-privkey.pem, and sw-cacert.pem to the /opt/cisco/controller/configuration/certs/ directory:

- **cd tls**
- **cp controller/ctl-privkey.pem controller/ctl-cert.pem /opt/cisco/controller/configuration/certs/**

- **cp of-switch/sw-cacert.pem /opt/cisco/controller/configuration/certs/**

b) Verify that these files were copied over:

- **cd /opt/cisco/controller/configuration/certs/**
- **ls -al**

c) Create the TLS keystore file.

1. Run the following commands:
   - **cd /opt/cisco/controller/configuration/certs/**
   - **cat ctl-privkey.pem ctl-cert.pem > server.pem**
   - **openssl pkcs12 -export -out server.p12 -in server.pem**
2. Enter and then verify an export password.
3. Run the **ls** command and verify that the following files are listed:
   - ctl-cert.pem
   - ctl-privkey.pem
   - server.p12
   - server.pem
   - sw-cacert.pem
4. Run the following command:

   /usr/java/jdk1.7.0_75/bin/keytool -importkeystore -srckeystore server.p12 -srcstoretype pkcs12 -destkeystore ctlKeyStore -deststoretype jks
5. Enter and then verify a destination keystore password.
6. Enter a source keystore password.

d) Create the TLS truststore file.

1. Run the following command:

   /usr/java/jdk1.7.0_75/bin/keytool -import -alias ca1 -file sw-cacert.pem -keystore ctlTrustStore
2. Enter and then verify a keystore password.

   At this point, the contents of the new certificate are displayed.
3. When prompted, enter **yes** to confirm that you want to trust this certificate.

e) Make the necessary edits to 42-openflowplugin.xml.

1. Navigate to the /opt/cisco/controller/etc/opendaylight/karaf/ directory.
2. In a text editor, open 42-openflowplugin.xml.
3. Make the following changes:

- Set the value of the transport-protocol parameter to `TLS`.
- Uncomment any parameters that are currently commented out, like the threads parameter.
- Change any instances of `CLASSPATH` to `PATH`.
- Set the correct absolute path for both the keystore and truststore parameters.
- Set values for the keystore-password, truststore-password, and certificate-password parameters.

f) Restart the controller:

- **sudo service monit stop**
- **sudo service controller stop**
- **sudo service controller start**
- **sudo service monit start**

Configuring TLS Support on a Nexus 3000 Series Switch

Step 1 (Optional) Open a console and run the following commands to delete the trustpoint and key that currently reside on the switch:

```
conf t
   crypto ca trustpoint myCA
   delete certificate force
   delete ca-certificate
   no rsakeypair myKey
   exit
no crypto ca trustpoint myCA
crypto key zeroize rsa myKey
```

Step 2 Set the hostname and domain name:

```
conf t
   hostname <device-name>
   ip domain-name cisco.com
```

Step 3 Create the trustpoint myCA and generate the key *myKey*.

```
crypto ca trustpoint myCA
crypto key generate rsa label myKey exportable modulus 2048
```

Step 4 Add the newly generated key to the trustpoint myCA:

```
crypto ca trustpoint myCA
   rsakeypair myKey
```

Step 5 Verify that the configuration was successful:

- **do show crypto ca trustpoints**
- **do show crypto key mypubkey rsa**

- **do show crypto ca certificates**

Step 6 Authenticate the trustpoint myCA.

a) From your TLS workspace, open the CA certificate:
cat mypersonalca/certs/ca.pem

b) Copy the certificate's text.

c) Run the following command:
crypto ca authenticate myCA

d) Paste the certificate text between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

Step 7 On the switch, generate the certificate request:

a) Run the following command:
crypto ca enroll myCA

b) When prompted, answer the questions with the responses provided in the following example:

```
Create the certificate request ..
 Create a challenge password. You will need to verbally provide this password to the CA
Administrator
in order to revoke your certificate. For security reasons your password will not be saved in the
configuration. Please make a note of it.
  Password:cisco123
 The subject name in the certificate will be the name of the switch.
 Include the switch serial number in the subject name? [yes/no]:no
 Include an IP address in the subject name [yes/no]:no
 Include the Alternate Subject Name ? [yes/no]:no
 The certificate request will be displayed...
-----BEGIN CERTIFICATE REQUEST-----
MIICtDCCAZwCAQAwIDEeMBwGA1UEAxMVbng3ay0xMS1vZnAuY2lzY28uY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA82
dEVgT3nv2v2tZ6jdSq4nDRWkhj9amj8u7sbOxhvpE1jQfWT6X7EE+mvM7/zZLwgdvLS/pMAmvO2jV18M61FiXMRD2xsYDVImzDC8PH4PV/
vcV0donqAD7kl+vFQRSAL/3JlfNpVQWtAGl1Uwi2di1a1qKUC8uAd8+QGcUlXWSmJreBicoioQJW31lWzoQImQZzIsO1fznTes9ychwVsSX
1iytC8r5KFjyniQ1iYAGghTrBmtrbSEo2PmEqfIPFCX/sEDhHOFnxCPNBtYC432PU5wIUKpSsyBuKZv78/S1gLNN9Aq/coeR9dhj0TEHzLX
5QaqknZyXPOn9RRRtIdwIDAQABoE8wFwYJKoZIhvcNAQkHMQoTCGNpc2NvMTIzMDQGCSqGSIb3DQEJDjEnMCUwIwYDVR0RAQH/BBkwF4IVb
ng3ay0xMS1vZnAuY2lzY28uY29tMA0GCSqGSIb3DQEBBQUAA4IBAQALglsuqWaNX0D97719psVmXwAc2ySZuEZ+RKli2Pi/Q1U+z7f2meyX
eyR2l3k86V6x1uAeAYERdy4Dp3cFPztMvmCHN23KOAEsVcwbbrePrxySfYrhR7/XTxP3jMHlKURKZTel0rZ/Cz3YDOtbCGJ6rmUp8/pPcO
GXifPPrUMCyGXtJrjoX5SvVTUJqNorNVztazcJRWUJ55hilSThlneDVHp6NHUGe98hm4GzlQ9qVbbgtS2KrMzZbw7xSqWnDORhwS7
sLnkYf+pdX3N0mw3wbD8uvhkzJBdN8jwxGHoadEBMc3gpv1OGnxZcnYW0o77txcod99Xootykri5aK+3R
-----END CERTIFICATE REQUEST-----
```

c) Copy the text of the certificate request (which you generated in the previous step).

d) On your TLS workspace, run the following command:
vi of-switch/sw-cert.req

e) Paste the text of the certificate request into sw-cert.req between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

Step 8 Generate the switch certificate:

a) Run the following command:
openssl ca -in of-switch/sw-cert.req -out of-switch/sw-cert.pem -config ./ca.cnf

b) When prompted, enter **y** to sign the certificate.

c) When prompted, enter **y** to commit the certificate request.

Step 9 Import the CA certificate to the switch:

a) Run the following command:
cat of-switch/sw-cert.pem

b) Copy the certificate's text.

c) Run the following command:
crypto ca import myCA certificate

d) Paste the certificate text between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

e) Verify the certificate was configured:
do show crypto ca certificates

Step 10 On the switch, enter the TLS Openflow configuration:

```
openflow
  switch 1
    protocol-version negotiate
    logging flow-mod
    tls trust-point local myCA remote myCA
    probe-interval 600
    pipeline 201
    controller ipv4 10.194.132.63 port 6653 vrf management security none
    controller ipv4 10.194.132.37 port 6653 vrf management security tls
    of-port interface ethernet1/49
    of-port interface ethernet1/50
hardware profile openflow
virtual-service n3kofa
  activate
```

Configuring TLS Support on a Catalyst 4000 Series Switch

Step 1 Clean before creating certificate and keys if already configured

```
conf t
   crypto key zeroize rsa myKey
end

conf t
   no crypto pki trustpoint myCA
end
```

Step 2 Set the hostname and domain name.

```
conf t
   hostname <device-name>
   ip domain-name cisco.com
end
```

Step 3 Set the switch's clock to a time and date that precedes the time and date set for the certificate.

The command you enter should look like the following example:

```
clock set 10:53:16 6 july 2015
```

Step 4 Create a public-private keypair on the switch:

```
conf t
crypto key generate rsa general label myKey exportable
```

Step 5 When prompted, enter **2048** as the size of the key modulus for your general purpose keys.

Step 6 Verify that the key was created:
do show crypto key mypubkey rsa

Step 7 Create the trustpoint and add the private key to it:

```
conf t
   crypto pki trustpoint myCA
   revocation-check none
   rsakeypair myKey
   enrollment terminal
   subject-name CN=swA
end
```

Step 8 View the trustpoint's status:
sh crypto pki trustpoint myCA status

Step 9 Create the switch's certificate signing request (CSR):

a) Run the following command:
crypto pki enroll myCA

b) When prompted, answer the questions with the responses provided in the following example:

```
% The subject name in the certificate will include: CN=swA
% The subject name in the certificate will include: cvg-cat4k-1.cisco.com
% Include the router serial number in the subject name? [yes/no]: NO
% Include an IP address in the subject name? [no]: no
Display Certificate Request to terminal? [yes/no]: yes
Certificate Request follows:
MIICmjCCAYICAQAwNDEMMAoGA1UEAxMDc3dBMSQwIgYJKoZIhvcNAQkCFhVjdmct
Y2F0NGstMS5jaXNjby5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCivYC4X72c4DRG7uWR6x+2UIxcq6TN4722yLPAFJb/7CXhCcvBNCMObuEuBaEJ
R9NUkwdGmvU87V4Fa4rsyTf/b19fAxcKE0FtRSk11LXp346gTHMCezEzkyRqpavl
ztB5cLrQfgzBwFgFm71q+48t+vsilWBpn0iCoiMcSMZC7+zY9yrBEcZCgGwIF5og
JePy+BxigtqfOQa2gwFOTbBfg53EUgW7Aq/3TTyyCjfwvNSBZEX9IFomPWQsl0eg
bcmHDf4R953H8/Iyfo0w3Jidiy8fMnKPWcmhfK+oUMY3q0UB/DmGu46yG/I82FWM
yB6KdJy/aFwY4Vybbwye6fvXAgMBAAGgITAfBgkqhkiG9w0BCQ4xEjAQMA4GA1Ud
DwEB/wQEAwIFoDANBgkqhkiG9w0BAQUFAAOCAQEAm0kqSfaCoiCrQoXihWcnd5zl
17mWPC15MLt2tJJn9g60otcUuqRiXwxCRQG0+k5Z3VVoGwA6ObQcJ+bKXMi8dh0k
sHBpQqV8o1PFIEbwnDoc2nRVhUOCqy+Vc/FMQxJztiK9n/j0emtyTI0fl3Ae1aSh
8X1y0tAi1u/F7T9/zyCRhNQhBTIeM717Ec0y5TMbaknoUrUwRoFwqtXgpE2tvyZq
m3y990Dunjr2yC40w6HmwCEgjfvFF1YY98MwhhhY7I+WLvwNFE96/dK49Nw2xqAC
YR4V4EAZVMEOcteSUoMnp4rcl63J75ZsnxcroRKlfp02E52+DR6VyALgWm+hjQ==

---End - This line not part of the certificate request---

Redisplay enrollment request? [yes/no]: no
```

Step 10 Create the switch's certificate:

a) Copy the text of the certificate request (which you generated in the previous step).

b) Run the following commands from the tls directory:

- **mkdir of-cat4k**
- **cd of-cat4k**
- **touch sw-cert.req**

c) Paste the text of the certificate request into sw-cert.req between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

Step 11 Verify that the request was made:
openssl asn1parse -in sw-cert.req

Step 12 Sign the switch's certificate:

a) Run the following command:
openssl ca -in of-cat4k/sw-cert.req -out of-cat4k/sw-cert.pem -days 3650 -notext -config ./ca.cnf

b) When prompted, enter **y** to sign the certificate.

c) When prompted, enter **y** to commit the certificate request.

Step 13 (Optional) View the contents of the switch's certificate:
openssl x509 -in of-cat4k/sw-cert.pem -text -purpose

Step 14 Import the CA and switch certificates to the router trustpoint:

a) On the switch, run the following commands:

- **cp mypersonalca/certs/ca.pem of-cat4k/sw-cacert.pem**
- **cat sw-cacert.pem**

b) Copy the certificate's text.

c) On the router, run the following command:
crypto pki authenticate myCA

d) Paste the certificate text you copied in Step 14b between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

e) After the line `-----END CERTIFICATE REQUEST-----`, enter **`quit`**.

f) When prompted, enter **`y`** to accept the certificate.

Step 15 Import the CA certificate to the switch.

a) On the switch, run the following command:
cat sw-cert.pem

b) Copy the certificate's text.

c) On the router, run the following command:
crypto pki import myCA certificate

d) Paste the certificate text you copied in Step 15b between the lines `-----BEGIN CERTIFICATE REQUEST-----` and `-----END CERTIFICATE REQUEST-----`.

e) After the line `-----END CERTIFICATE REQUEST-----`, enter **`quit`**.

Step 16 (Optional) Verify that both the switch and CA certificates are present:
do show crypto pki cert

Step 17 On the swtich, enter the TLS OpenFlow configuration:

a) Enter the following configuration information:

```
openflow
  switch 1
   pipeline 1
   of-port interface TenGigabitEthernet1/1
   of-port interface TenGigabitEthernet1/2
   logging flow-mod
   protocol-version negotiate
   controller ipv4 <controller1-IP-address> port 6653 vrf mgmtVrf security none
   controller ipv4 <controller2-IP-address> port 6653 vrf mgmtVrf security tls
   tls trust-point local myCA remote myCA
end
```

b) Verify that the 2 controllers you just configured are listed:
sh openflow switch 1 controllers

Web Server Certificate Installation

If your company has a pre-signed certificate file, you can use that instead of the certificate file that comes with Open SDN Controller.Before you complete the following procedure, make sure that your certificate's .crt and .key files are available.

Step 1 Copy your certificate's .crt file.

Step 2 On the machine on which Open SDN Controller is installed, navigate to the following directory:
/etc/pki/tls/certs/

Step 3 Overwrite ca.crt with your certificate's .crt file, ensuring that the filename remains ca.crt.

Step 4 Copy your certificate's .key file.

Step 5 Navigate to the following directory:
/etc/pki/tls/private/

Step 6 Overwrite ca.key with your certificate's .key file, ensuring that the filename remains ca.key.

Step 7 Restart the HTTP service by running the following command:
sudo service httpd restart

Port Usage Table

The following table lists the ports used by Open SDN Controller (in both single node and 3-node cluster setups) and their purpose. When viewing this table, note that:

- All of the ports listed below are configured to use TCP except for port 53, which uses UDP.
- Any available port can used for outgoing traffic.

- In 3 –node cluster setups, any available port can be used to transfer data between those three nodes.

Table 2: Ports Used by Open SDN Controller

Port Number	Purpose
22	Incoming and outgoing SSH traffic
53	Outbound DNS traffic
80	Incoming HTTP traffic
123	NTP connections
179	Southbound BGP connections
443	Incoming and outgoing HTTPS traffic
830	Southbound NETCONF connections
1099	Remote JMX connections
4189	Southbound PCEP connections
6633	Southbound OpenFlow connections
6653	Southbound OpenFlow connections
44444	Remote JMX connections

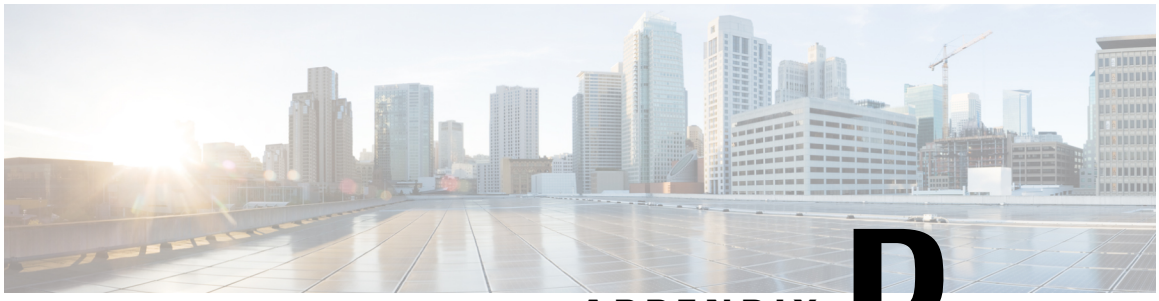
Supported Protocols and Services

The following table lists the protocols, TCP/IP services, and platform system services that Open SDN Controller supports.

Table 3: Protocols and Services Supported by Open SDN Controller

Protocols	
BGP-LS/PCEP	NETCONF
ICMP	OpenFlow
TCP/IP Services	
DNS	NTP
HTTPS	SSH

JMX	
Platform System Services	
cassandra	flume
collectd	httpd
controller	Java
cyanite	pathman
elasticsearch	



APPENDIX D

NETCONF

The following topics provide an overview of NETCONF and describes common tasks you would perform for the NETCONF-enabled devices in your network:

Overview

The Network Configuration Protocol (NETCONF) defines a simple mechanism through which a network device can be managed, configuration data can be retrieved, and new configuration data can be uploaded and manipulated. NETCONF uses Extensible Markup Language (XML)-based data encoding for the configuration data and protocol messages.

Open SDN Controller operates as both a NETCONF server and client. As a server, the controller manages general network communication and processes remote procedure calls (RPCs). And as a client, the controller connects to NETCONF-enabled devices and manages them through the NETCONF connector. The following sections will cover typical tasks that are completed for NETCONF-enabled devices.

Mounting NETCONF Devices to the Controller

Before you proceed with Step 1, you must generate a security token. See Making RESTCONF Requests for more information.

Step 1 Make the following POST request:
URL—https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/controller-config/yang-ext:mount/config:modules

Specify values for the following parameters in the request's payload:

- Name
- Address
- Port
- Username
- Password

In the following example, a device named *asr9k-1-netconf* with an IP address of 192.0.2.116 is mounted to the controller.

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:opendaylight:params:xml:ns:yang:controller:config">
 <type
xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">prefix:sal-netconf-connector</type>

 <name>asr9k-1-netconf</name>
 <address
xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">192.0.2.116</address>

 <port xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">830</port>
 <username
xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">cisco</username>
 <password
xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">cisco</password>
 <tcp-only
xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">false</tcp-only>
 <event-executor xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">
  <type
xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:netty">prefix:netty-event-executor</type>

  <name>global-event-executor</name>
 </event-executor>
 <binding-registry xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">
  <type
xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:binding">prefix:binding-broker-osgi-registry</type>

  <name>binding-osgi-broker</name>
 </binding-registry>
 <rpc-registry xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">
  <type
xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:binding">prefix:binding-rpc-registry</type>

  <name>binding-rpc-broker</name>
 </rpc-registry>
 <dom-registry xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">
  <type
xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:md:sal:dom">prefix:dom-broker-osgi-registry</type>

  <name>dom-broker</name>
 </dom-registry>
 <client-dispatcher xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">
```

```
  <type
xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:config:netconf">prefix:netconf-client-dispatcher</type>

  <name>global-netconf-dispatcher</name>
 </client-dispatcher>
<processing-executor xmlns="urn:opendaylight:params:xml:ns:yang:controller:md:sal:connector:netconf">

  <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:threadpool">
prefix:threadpool</type>
  <name>global-netconf-processing-executor</name>
 </processing-executor>
</module>
```

Step 2 Make the following GET request to determine the device's connection status and view information such as its serial number and installed software:
URL—https://token:$token@<*controller-IP-address*>/controller/restconf/operational/opendaylight-inventory:nodes/node/<*device-hostname*>

If the router is currently connected, its entry will display `"netconf-node-inventory:connected": true,`.

Note the following:

- The Open SDN Controller will use the router name you specify to identify that router.
- The IP address, username, and password you specify should already be configured on the router.
- The IP address you specify should be reachable from the controller.
- You may need to connect to a remote NETCONF device that does not support NETCONF monitoring schema. For connection instructions, view the "Connecting to a device not supporting netconf monitoring" topic here.

Viewing the APIs Supported by a Mounted Device

Step 1 Access the API documentation by selecting **Available APIs** from the toolbar's Help menu.

Step 2 Click the Mounted Resources tab to view a listing of all the devices mounted on the controller.

Step 3 Click the link for the device you want to view API information for.
For more information about APIs, see Viewing RESTCONF API Documentation.

Modifying Mounted Device Configuration Parameters

After a NETCONF device has been mounted to the controller and a connection is established, you can change the settings for any of its configuration parameters at runtime. In this example, say you want to modify the username and password configured for the device we mounted previously in this section. To do so, make a

POST request using the following URL, specifying the new username and password values in the request's payload:

https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/controller-config/yang-ext:mount/config:modules

Note the following:

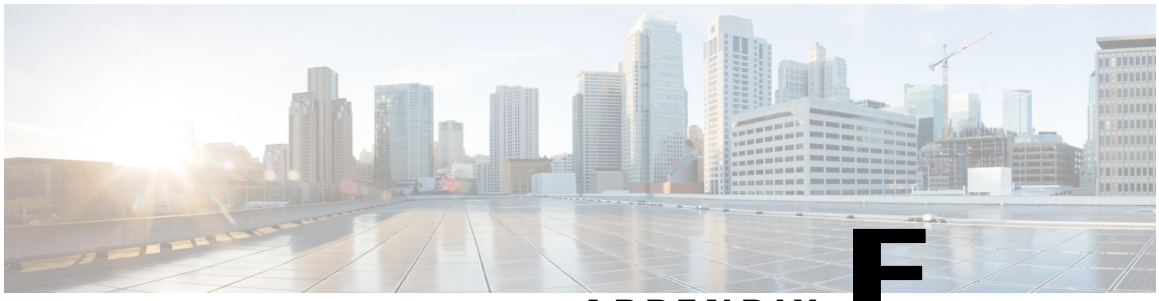
- Before you make this request, you must first generate a security token. See Making RESTCONF Requests for more information.
- To view sample NETCONF configurations, visit the following URL:

  https://wiki.opendaylight.org/view/OpenDaylight_Controller:Config:Examples:Netconf

Deleting a Mounted Device

To delete a mounted device, make a DELETE request using the following URL, in which you provide the controller's IP address and the mounted device's hostname:

https://token:$token@<*controller-IP-address*>/controller/restconf/config/opendaylight-inventory:nodes/node/controller-config/yang-ext:mount/config:modules/module/odl-sal-netconf-connector-cfg:sal-netconf-connector/<*device-hostname*>



APPENDIX E

Adding a New Application to the OSC UI

The OSC UI is a Karaf-based UI platform that allows you to create new applications and install those applications within the OSC UI via the OSGi Blueprint Container specification. Applications consist of the following components, which are typically packaged as separate Maven JAR files:

- Application module—Contains all of the JS and HTML code for an application.
- Application bundle—Contains an application's configuration file (blueprint.xml), which is read by the Karaf container and used to deploy that application within the OSC UI. The application bundle embeds the contents of the corresponding application module to ensure that they can be accessed from a browser. Note that only the application bundle, and not the application module, is deployed in Karaf.

You also have the option of packaging an application's module and bundle files together in one Maven JAR file. By doing so, you will not need to embed the module's content within the bundle.

To add a new application to the OSC UI, you will need to do the following:

1. Create a new JavaScript module.
2. Add a new OSGi Blueprint bundle.
3. Add a new Karaf feature for your application.

Before you proceed, note the following:

- You can deploy your application's installer (.kar) file from the Features page. See Installing New Features for more information.
- We recommend that you use both angularJS and requireJS when writing the code for your application.
- The following topics assume that you have a basic understanding of Karaf. Refer to the Karaf Developer's Guide for more information.
- To learn more about Blueprint, click here.
- OSC and its UI run on separate Karaf instances.

Creating a New JavaScript Module

The first thing you need to do when developing a new OSC UI application is create a JavaScript module. This module should be coded using angularJS and requireJS and packaged as a Maven JAR file. The structure for your project should look similar to the following example:

```
<module-name>-resources
-- src
---- main
------resources
-------- <module-name>
---------- <module-name>.module.js
---------- <module-name>.controller.js
---------- <module-name>.services.js
---------- <module-name>.directives.js
---------- <module-name>.filter.js
---------- index.tpl.html
---------- <module-stylesheet>.css
-- pom.xml
```

Note that all of the code for your module will reside under your project's resources folder.

To create a new JavaScript module, do the following:

1. Define the module.
2. Create the necessary module components.
3. Edit the module's POM files.

Defining the Module

Defining your JavaScript module is a five-step process that involves the following tasks:

1. Creating your module's JavaScript file.
2. Setting the Register function.
3. Setting the route.
4. Adding the module to the navigation menu.
5. Linking to the controller file.

Creating Your Module's JavaScript File

Create a new file and save it with a name such as topology.module.js. The following example illustrates the contents of a standard module.js file:

```
define(['angularAMD','app/routingConfig', 'angular-ui-router','app/core/core.services'],
function(ng) {
    var module = angular.module('app.a_module', ['ui.router.state', 'app.core']);

    // module configuration
    module.config(function() {
        [...]
    });

   return module;
});
```

In this example, the angularJS module is surrounded by a define function. This allows requireJS to see our module.js files. The first argument of the define function is an array which contains all of the module dependencies. The second argument is a callback function whose body contains the angularJS module code. The function parameters correspond with the order of dependencies, and each dependency is inserted into a parameter (if it is provided). Finally, the angular module is returned in order to enable its insertion as a parameter in any other modules you create.

For each new module, you must have at least two dependencies:

- angularAMD—This is a angularJS wrapper that provides Asynchronous Module Definition (AMD) support, which is used by requireJS. For more information, click here.
- app/core/core.services—This dependency is mandatory if you want to add content to the navigation menu, the left bar, or the top bar.

The following dependencies are not mandatory but are used often:

- angular-ui-router—A library that provides URL routing.
- routingConfig—Sets level access to a page.

Setting the Register Function

If your module is required by the main application, you need to register your angular components because the parent OSC UI application will be already be bootstrapped. The OSC UI will not see your components at runtime unless you add the following code.

Note If your module is only used by another module, you can skip this step.

```
module.config(function($compileProvider, $controllerProvider, $provide) {
    module.register = {
      controller : $controllerProvider.register,
      directive : $compileProvider.directive,
      factory : $provide.factory,
      service : $provide.service

    };
```

Setting the Route

Next, set up your module's route by adding the $stateProvider parameter to your module's configuration method.

```
module.config(function($stateProvider) {
    var access = routingConfig.accessLevels;
    $stateProvider.state('main.module', {
      url: 'module',
      views : {
        'content' : {
          templateUrl: 'src/app/module/module.tpl.html',
          controller: 'ModuleCtrl'
        }
      }
    });
});
```

Adding the Module to the Navigation Menu

In order to add an item to the navigation menu, the NavMenuHelper parameter in your module's configuration method must be set, as illustrated in the following example. The first parameter is an ID that refers to a level of your menu and the second parameter is an object.

```
var module = angular.module('app.a_module', ['app.core']);
module.config(function(NavMenuHelper) {
    NavMenuHelper.addToMenu('myFirstModule', {
        "link" : "#/module/index",
        "active" : "module",
        "title" : "My First Module",
        "icon" : "icon-sitemap",
        "page" : {
            "title" : "My First Module",
            "description" : "My first module"
        }
    });
});
```

Currently, two levels of ID parameter support are provided. For example, if your module's ID is *rootNode.childNode*, the helper will look for a node named *rootNode* and append it with *childNode*. If the root node does not exist, it will create it automatically.

Linking to the Controller File

To link to the controller file, use the NavHelperProvider. It contains a method that will load the specified file.

```
[...]
NavHelperProvider.addControllerUrl('<path-to-module-folder>/
<module-name>.controller');
```

Setup of the module.js file is now complete.

Creating Necessary Components

The process for creating the controller and other necessary components is similar to that for defining your module.

1. Add the module definition.
2. Specify the relative path to the module definition.
3. Create your methods using angularJs.

   In the following example, we are setting up the register controller module:

```
define(['<relative-path-to-module>/<module-name>.module'], function(module) {
    module.register.controller('ModuleCtrl', function($rootScope, $scope) {
    });
});
```

Remember that you don't need to register your angular components if your module only refers to another module.

Editing POM Files

The last thing you need to do to create a new JavaScript module is modify the POM.xml files that reside in the main OSC UI directory (dlux/), the modules directory (dlux/modules/), and the dlux-web directory (dlux/dlux-web/). Edit the files as follows:

Note If you are writing an application that will reside outside of the OSC UI repository, you do not need to make the changes described in this topic.

POM.xml File in the dlux/ Directory

```
<properties>
     <nexus.repository.release>opendaylight.release
     </nexus.repository.release>
     <nexus.repository.snapshot>opendaylight.snapshot
     </nexus.repository.snapshot>
     <application-name.resources.version><Version>
     </application-name.resources.version>
     ...........
<properties>
```

POM.xml File in the dlux/modules/ Directory

```
<modules>
    <module>{application-directory-name}</module> //For example "grouppolicy-resources" or
 "loader-resources"
    .......
</modules>
```

POM.xml File in the dlux/dlux-web/ Directory

```
<dependencies>
        <dependency>
           <groupId>org.opendaylight.dlux</groupId>
           <artifactId>dlux.{application-name}.resources</artifactId>
           <version>${{application-name}.resources.version}</version>
        </dependency>
    </dependencies>
      ......
    <includeArtifactIds>    //Line 183
        dlux.{application-name}.resources    //for example "dlux.grouppolicy.resources" or
 "dlux.topology.resources"
        .....
    </includeArtifactIds>
```

Adding a New OSGi Blueprint Bundle

The OSGi Blueprint Container specification allows you to use dependency injection in your OSGi environment. Each OSC UI application module registers itself via its Blueprint configuration. Each application will have its own blueprint.xml file in which to place its configuration.

1 Create a Maven project with the following structure to place you Blueprint configuration:

```
AppModuleName
   src
     main
       resources
         OSGI-INF
           blueprint
             blueprint.xml
   pom.xml
```

2 In the pom.xml file, add a Maven plug-in to unpack your module's code under this project's generated-resources directory.

The following sample POM file is provided for your reference.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.opendaylight.dlux</groupId>
        <artifactId>bundles</artifactId>
        <version>0.3.0-SNAPSHOT</version>
        <relativePath>../</relativePath>
    </parent>
    <groupId>org.opendaylight.dlux</groupId>
    <artifactId>dlux.topology</artifactId>
    <packaging>bundle</packaging>
    <dependencies>
        <dependency>
            <groupId>org.osgi</groupId>
            <artifactId>org.osgi.core</artifactId>
        </dependency>
        <dependency>
            <groupId>org.osgi</groupId>
            <artifactId>org.osgi.compendium</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.felix</groupId>
            <artifactId>org.osgi.compendium</artifactId>
            <version>${apache.felix.osgi.compendium.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
        </dependency>
        <dependency>
            <groupId>org.opendaylight.dlux</groupId>
            <artifactId>loader</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.opendaylight.dlux</groupId>
            <artifactId>dlux.topology.resources</artifactId>
            <version>${topology.resources.version}</version>
        </dependency>
    </dependencies>
    <build>
        <resources>
            <resource>
                <directory>target/generated-resources</directory>
            </resource>
            <resource>
                <directory>src/main/resources</directory>
            </resource>
        </resources>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <version>2.6</version>
                <executions>
                    <!--loader Resources-->
                    <execution>
                        <id>unpack-loader-resources</id>
                        <goals>
                            <goal>unpack-dependencies</goal>
                        </goals>
                        <phase>generate-resources</phase>
                        <configuration>
                            <outputDirectory>${project.build.directory}/
generated-resources</outputDirectory>
                            <groupId>org.opendaylight.dlux</groupId>
```

```
<includeArtifactIds>dlux.topology.resources</includeArtifactIds>
                            <excludes>META-INF\/**</excludes>
                            <excludeTransitive>true</excludeTransitive>
                            <ignorePermissions>false</ignorePermissions>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.felix</groupId>
                <artifactId>maven-bundle-plugin</artifactId>
                <extensions>true</extensions>
                <configuration>
                    <instructions>
                        <Import-Package>org.osgi.service.http,
                            org.osgi.framework;version="1.0.0",
                            org.opendaylight.dlux.loader,
                            org.slf4j
                        </Import-Package>
                        <Export-Package></Export-Package>
                    </instructions>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <scm>
<connection>scm:git:ssh://git.opendaylight.org:29418/
dlux.git</connection>
<developerConnection>scm:git:ssh://git.opendaylight.org:29418/dlux.git</developerConnection>

        <tag>HEAD</tag>
        <url>https://wiki.opendaylight.org/view/OpenDaylight_dlux:Main</url>
    </scm>
</project>
```

Since your bundle will eventually be deployed in Karaf as a feature, your bundle should contain all of your module's code. You should not encounter any problems if you choose to combine both the module and bundle together into one project.

3 Create the blueprint.xml configuration file in the src/main/resources/OSGI-INF/blueprint directory.

When creating a new application's configuration file, ensure that it is formatted in a similar fashion to the following example:

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
 <reference id="httpService" availability="mandatory" activation="eager"
interface="org.osgi.service.http.HttpService"/>
    <reference id="loader" availability="mandatory" activation="eager"
interface="org.opendaylight.dlux.loader.DluxModuleLoader"/>

    <bean id="bundle" init-method="initialize" destroy-method="clean"
class="org.opendaylight.dlux.loader.DluxModule">
      <property name="httpService" ref="httpService"/>
      <property name="loader" ref="loader"/>
      <property name="moduleName" value="topology "/>
      <property name="url" value="/src/app/topology"/>
      <property name="directory" value="/topology"/>
      <property name="requireJs" value="app/topology/topology.module"/>
      <property name="angularJs" value="app.topology"/>
       <property name="cssDependencies">
            <list>

<value>http://yui.yahooapis.com/3.18.1/build/cssreset/cssreset-min.css</value>
                 <value>src/app/topology/topology-custom.css</value>
            </list>
        </property>
    </bean>
</blueprint>
```

In the configuration above, two references with IDs are listed: httpService and loader. These two beans will have already been initialized by dlux-core, so any new application can use them. Without these two bean references, a new application will not be able to register.

4 Initialize your application bean, which will be an instance of class org.opendaylight.dlux.loader.DluxModule.

In addition to httpService and loader, there are 6 properties that you should specify for this bean:

- *moduleName*—Name of your module. This name should be unique in the OSC UI.
- *url*—This is the URL requireJS will use to load your module's .js and .html files into the OSC UI. This is also the URL that a browser will use to load static .html, .js, and .css files. Since requireJS in the OSC UI has a base path of src, every URL you specify for this step should start with /src so that requireJS and browsers can access the appropriate files.
- *directory*—In your bundle's pom.xml file, you unpack your module's code. The directory you specify here is where your actual static files reside. The URL you specified in the previous bullet is registered with httpService, so when a browser makes a call to that URL, it will be redirected to the directory specified here.
- *requireJS*—The path to your requireJS module. If you look closely in the previous example, you will see that the initial path of requireJS app/topology matches with the last part of the URL. This is the path that will be used by requireJS.
- *angularJS*—Name of your angularJS module.
- *cssDependencies*—If your application has any internal or external CSS dependencies, then those can be added here. If you create your own .css files, point to those files.

After you deploy your bundle in Karaf, Karaf will read your application's blueprint.xml file and register the application with the OSC UI. Once successful, refresh the OSC UI and you will see your application in the Applications pane.

Adding a New Karaf Feature for Your Application

At this point, you have written your JavaScript code and created a bundle that Karaf can understand. The final step is to test and deploy your bundle. Before you proceed, ensure that the odl-dlux-core feature is already enabled in Karaf.

1 Copy your bundle JAR file and place it in the deploy directory of your Karaf-based controller.

2 From the Karaf Console, install your bundle:

root@karaf> bundles:install -s mvn:mvn:org.opendaylight.dlux/dlux.topology/0.3.0

You may want to create your own Karaf feature in a production environment, which may deploy one or more bundles. All the Karaf-based features in the OSC UI are defined in the features.xml file, which can be found in the features/src/main/resources/ directory. A standard feature definition can have a dependency on another feature and one or more bundles, as is the case in the following example:

```
<feature name="odl-dlux-node" version='${project.version}' description="Enable nodes in
Opendaylight dlux">
        <feature>odl-dlux-core</feature>
        <bundle>mvn:org.opendaylight.dlux/dlux.node/${project.version}</bundle>
 </feature>
```

If you are updating code in the OSC UI repository, you can update the existing features.xml file. If your project resides outside of the repository, you can create a new feature there that includes your application bundle and has a dependency on the odl-dlux-core feature.