CONFIGURATION GUIDE

# Extreme SLX-OS SDN Configuration Guide, 16r.1.01

## Supporting the ExtremeRouting SLX 9850 Router

*53-1004813-02*
*15 June 2018*

Notice: This document is for informational purposes only and does not set forth any warranty, expressed or implied, concerning any equipment, equipment feature, or service offered or to be offered by Brocade. Brocade reserves the right to make changes to this document at any time, without notice, and assumes no responsibility for its use. This informational document describes features that may not be currently available. Contact a Brocade sales office for information on feature and product availability. Export of technical data contained in this document may require an export license from the United States government.

The authors and Brocade Communications Systems, Inc. assume no liability or responsibility to any person or entity with respect to the accuracy of this document or any loss, cost, liability, or damages arising from the information contained herein or the computer programs that accompany it.

The product described by this document may contain open source software covered by the GNU General Public License or other open source license agreements. To find out which open source software is included in Brocade products, view the licensing terms applicable to the open source software, and obtain a copy of the programming source code, please visit http://www.brocade.com/support/oscd.

# Contents

# Preface

# Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Extreme technical documentation.

## Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION
**A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.**

DANGER
***A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.***

## Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

| Format | Description |
|---|---|
| **bold** text | Identifies command names. |
| | Identifies keywords and operands. |
| | Identifies the names of GUI elements. |
| | Identifies text to enter in the GUI. |
| *italic* text | Identifies emphasis. |
| | Identifies variables. |
| | Identifies document titles. |
| `Courier font` | Identifies CLI output. |

| Format | Description |
|---|---|
| | Identifies command syntax examples. |

## Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

| Convention | Description |
|---|---|
| **bold** text | Identifies command names, keywords, and command options. |
| *italic* text | Identifies a variable. |
| [ ] | Syntax components displayed within square brackets are optional.<br>Default responses to system prompts are enclosed in square brackets. |
| { **x** \| **y** \| **z** } | A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options. |
| **x** \| **y** | A vertical bar separates mutually exclusive elements. |
| < > | Nonprinting characters, for example, passwords, are enclosed in angle brackets. |
| ... | Repeat the previous element, for example, *member*[*member*...]. |
| \ | Indicates a “soft” line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash. |

# Extreme resources

Visit the Extreme website to locate related documentation for your product and additional Extreme resources.

White papers, data sheets, and the most recent versions of Extreme software and hardware manuals are available at www.extremenetworks.com. Product documentation for all supported releases is available to registered users at www.extremenetworks.com/support/documentation.

# Document feedback

Quality is our first concern at Extreme, and we have made every effort to ensure the accuracy and completeness of this document. However, if you find an error or an omission, or you think that a topic needs further development, we want to hear from you.

You can provide feedback in two ways:

- Use our short online feedback form at http://www.extremenetworks.com/documentation-feedback-pdf/
- Email us at internalinfodev@extremenetworks.com

Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

# Contacting Extreme Technical Support

As an Extreme customer, you can contact Extreme Technical Support using one of the following methods: 24x7 online or by telephone. OEM customers should contact their OEM/solution provider.

If you require assistance, contact Extreme Networks using one of the following methods:

- GTAC (Global Technical Assistance Center) for immediate support
  - Phone: 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact.
  - Email: support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.
- GTAC Knowledge - Get on-demand and tested resolutions from the GTAC Knowledgebase, or create a help case if you need more guidance.
- The Hub - A forum for Extreme customers to connect with one another, get questions answered, share ideas and feedback, and get problems solved. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
- Support Portal - Manage cases, downloads, service contracts, product licensing, and training and certifications.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

# About This Document

## Supported hardware and software

In those instances in which procedures or parts of procedures documented here apply to some devices but not to others, this guide identifies exactly which devices are supported and which are not.

Although many different software and hardware configurations are tested and supported by Extreme Networks, Inc. for SLX-OS Release 16r.1.01, documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release:

- ExtremeRouting SLX 9850-4
- ExtremeRouting SLX 9850-8

# OpenFlow 1.3

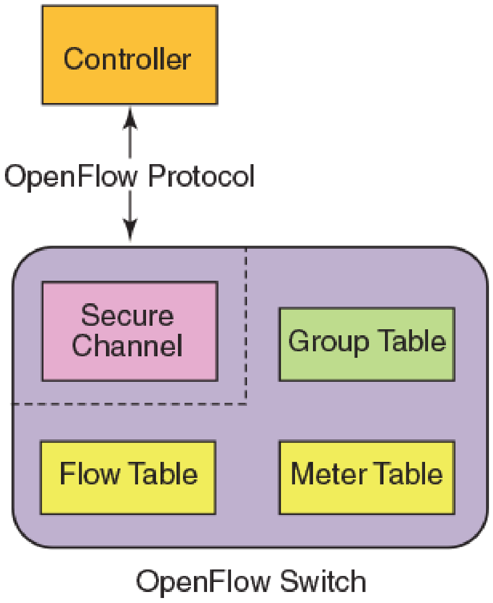
## OpenFlow 1.3 protocol overview

An OpenFlow-enabled switch supports an OpenFlow Client (control plane software), which communicates with an OpenFlow Controller using the OpenFlow protocol. The OpenFlow Controller runs on a server or a server cluster. OpenFlow-enabled routers support the abstraction of a flow table, which is manipulated by the OpenFlow Controller. The flow table contains flow entries. Each flow entry represents a flow (that is, packets with a given MAC address, VLAN tag, IP address, or TCP/UDP port, and so on). The flow table is sorted by flow priority, which is defined by the OpenFlow Controller. The highest priority flows are at the top of the flow table.

Incoming packets on an OpenFlow-enabled port are matched (in order of priority) against the flow entries defined for that port by the OpenFlow Controller. If the packet matches a given flow entry, the flow-matching process stops, and the set of actions defined for that flow entry are performed. Packets that do not match any flow entry are dropped by default. The implementation of OpenFlow supports an option to send such packets to the OpenFlow Controller.

An OpenFlow switch maintains one flow table, which is used for packet processing. The switch performs the actions listed in the table entry corresponding to the matched flow. The OpenFlow Controller manages the OpenFlow switch using the OpenFlow protocol. The OpenFlow Controller can add, delete, or modify flows and get statistics for ports and flows and other information using the OpenFlow protocol.

FIGURE 1 OpenFlow 1.3 architecture



OpenFlow 1.3 defines three types of tables:

- Flow table
- Group table
- Meter table

## Flow table entries

Only one flow table is supported on an OpenFlow port. Each flow table entry contains the fields described in the following table.

TABLE 1 Flow table entries

| Field | Description |
|---|---|
| Match fields | The match fields consist of ingress ports, packet header fields |
| Priority | Matching precedence of the entry |
| Counters | Statistics for matching packets |
| Instructions | Action set or pipeline processing |
| Cookie | Opaque data sent by the OpenFlow Controller |

The following match fields are supported:

- All Layer 2 header fields
- All Layer 3 header fields
- All Layer 2 and IPv4 header fields jointly

**TABLE 2** OpenFlow match fields supported is SLX-OS

| Match field | Prerequisite | Description |
|---|---|---|
| OXM_OF_IN_PORT | None | Ingress port. Numerical representation of incoming port, starting at 1. This may be a physical or switch-defined logical port. |
| OXM_OF_ETH_SRC | None | Ethernet source MAC address |
| OXM_OF_ETH_DST | None | Ethernet destination MAC address |
| OXM_OF_ETH_TYPE | None | Ethernet type of the packet |
| OXM_OF_VLAN_VID | None | VLAN-ID from 802.1Q header |
| OXM_OF_VLAN_PCP | VLAN_VID = None | VLAN priority (VLAN-PCP) from 802.1Q header |
| OXM_OF_IPV4_SRC | Ether type = 0x0800 | IPv4 source address |
| OXM_OF_IPV4_DST | Ether type = 0x0800 | IPv4 destination address |
| OXM_OF_IPV6_SRC | Ether type = 0x86dd | IPv6 source address |
| OXM_OF_IPV6_DST | Ether type = 0x86dd | IPv6 destination address |
| OXM_OF_IP_PROTO | Ether type = 0x0800 | IPv4 protocol number |
| OXM_OF_IP_DSCP | Ether type = 0x0800 | IPv4 DSCP (IPv4 ToS bits) |
| OXM_OF_TCP_SRC | IP PROTO = 6 | TCP source port |
| OXM_OF_TCP_DST | IP PROTO = 6 | TCP destination port |
| OXM_OF_UDP_SRC | IP PROTO = 17 | UDP source port |
| OXM_OF_UDP_DST | IP PROTO = 17 | UDP destination port |

# OpenFlow instructions

Each flow entry has a set of instructions that are executed when the packet matches the entry.

The instruction set associated with each flow entry can have a maximum of one instruction of each type. The following table shows the actions supported on different devices.

NOTE
Only one flow table is supported on all platforms for OpenFlow instructions.

**TABLE 3** Actions for flow table instruction

| Actions | Description | SLX 9850-4 | SLX 9850-8 |
|---|---|---|---|
| Write-Action actions ( Req ) | Adds or overwrites specified actions to the action set. | Yes | Yes |
| Apply-Action actions | Applies the specified actions immediately. | Yes | Yes |
| Clear-Action actions | Clears all the actions in the action set. | No | No |
| Meter *meter-id* | Directs the packet to the specified meter. | Yes | Yes |
| Goto -Table *next-table-id* ( Req ) | Indicates the next table in pipeline processing. | No* | No* |
| Write-Metadata metadata/mask | Writes the metadata field from the mask. | No* | No* |

NOTE
*: Not required with one flow table.

# OpenFlow actions

Each flow has a set of instructions that are executed when the packet matches the flow as per OpenFlow 1.3 specifications. Each flow can have a maximum of one instruction of each type.

A switch can reject a flow entry if the switch is unable to execute the instructions associated with the flow entry. In this case, the switch returns an unsupported flow error. Flow table may not support every match, every instruction, or every action.

**TABLE 4** Supported actions for different devices

| Actions | Description | SLX 9850-4 | SLX 9850-8 |
|---|---|---|---|
| Output (Req) | Forwards the packet to a specified OpenFlow port. If out-port is Controller, then the packet will be sent as packet-in message. | Yes | Yes |
| Drop (Req) | No explicit drop action. Packet with empty action set should be dropped. | Yes | Yes |
| Group | Processes the packet through the specified group. | Yes | Yes |
| Set field | Modifies the values of the packet header based on the field type. | Yes ( VLAN_VID, VLAN-PCP) | Yes ( VLAN_VID, VLAN-PCP) |
| Push-Tag/ Pop-Tag | Adds and removes tag (newly inserted tags are always the outermost tags). | Yes | Yes |
| Set-Queue | Set the queue ID for the packet. | Yes | Yes |
| Change TTL | Modify the TTL value. | No | No |

Set field is used to modify the packet content. Devices support VLAN ID and VLAN-PCP modifications only.

- OXM_OF_VLAN_VID modifies the outermost VLAN ID, when tag-type is 0x8100. In case of untagged packet, a new VLAN header is created.
- OXM_OF_VLAN_PCP modifies the PCP value in the outermost VLAN header. This action is ignored for untagged packet.

## *OpenFlow actions output*

The OpenFlow flow supports the following Openflow ports: physical, logical, and reserved.

**TABLE 5** Supported actions output on device

| Type | On device |
|---|---|
| Hardware interface of the switch | Yes |
| LAG | No |
| Tunnel | No |
| All | No |
| Controller | Yes |
| Table | No |
| In_port | No |
| Local | No |
| Normal | No |
| Flood | No |

Flow action supports more than one OpenFlow physical port in the action-list.

### Limitations

- Combination of physical and reserved ports are not supported by flow action.
- Controller port is rate limited to 4 kbps from hardware to CPU.

## OpenFlow actions push and pop VLAN

The following actions and limitations are supported for push and pop VLAN.

Push VLAN adds a VLAN tag to the existing packet. The flow configuration supports only 0x8100 tag-type, push VLAN with any other tag-types are not supported. Push VLAN is only valid on physical port.

Pop VLAN action removes the outermost VLAN header. If the action-list has both pop VLAN and set field greater than VLAN_VID or push VLAN, then the flow is rejected. Pop VLAN is used only with physical port out port. Multiple pop VLAN action in the same flow is not supported.

## OpenFlow TCAM profiles

The following table illustrates the flow match in the different OpenFlow TCAM profiles.

**TABLE 6** Flows and TCAM profiles

| OpenFlow features | | Default profile | Optimized profile 1 | Optimized profile 2 |
|---|---|---|---|---|
| Match | In port | Yes | Yes | Yes |
| | VLAN ID | Yes | Yes | Yes |
| | VLAN-PCP | Yes | Yes | Yes |
| | Source MAC address | No | Yes | Yes |
| | Destination MAC address | No | Yes | Yes |
| | Ether type = 0x0800 | Yes | Yes | Yes |
| | Ether type except 0x0800 | No | Yes | Yes |
| | Source IP address | Yes | Yes | Yes |
| | Destination IP address | Yes | Yes | Yes |
| | DSCP | Yes | Yes | Yes |
| | Layer 4 source port | Yes | Yes | Yes |
| | Layer 4 destination port | Yes | Yes | Yes |
| Action | Single out port | Yes | Yes | Yes |
| | Multiple out port | Yes | Yes | Yes |
| | Send to controller | Yes | Yes | Yes |
| | Drop | Yes | Yes | Yes |
| | VLAN modification (set/push/pop) with single out port | No | Yes | No |
| | VLAN modification (set/push/pop) with multiple out port | Yes | Yes | Yes |
| | VLAN-PCP modification with single and multiple out ports | No | Yes | No |
| | Flow statistics | Yes | Yes | Yes |
| | Meter | No | No | Yes |
| | Enqueue | No | No | Yes |
| | Group (all group types) | Yes | Yes | Yes |

## Supported OpenFlow messages

The following OpenFlow messages are supported on the devices.

**TABLE 7** OpenFlow messages

| Message type | SLX 9850-4 | SLX 9850-8 |
|---|---|---|
| OFPT_HELLO | Yes | Yes |
| OFPT_ERROR | Yes | Yes |
| OFPT_ECHO_REQUEST | Yes | Yes |
| OFPT_ECHO_REPLY | Yes | Yes |
| OFPT_EXPERIMENTER | No | No |
| OFPT_FEATURES_REQUEST | Yes | Yes |
| OFPT_FEATURES_REPLY | Yes | Yes |
| OFPT_GET_CONFIG_REQUEST | No | No |
| OFPT_GET_CONFIG_REPLY | No | No |
| OFPT_SET_CONFIG | No | No |
| OFPT_PACKET_IN | Yes | Yes |
| OFPT_FLOW_REMOVED | Yes | Yes |
| OFPT_PORT_STATUS | Yes | Yes |
| OFPT_PACKET_OUT | Yes | Yes |
| OFPT_FLOW_MOD | Yes | Yes |
| OFPT_GROUP_MOD | Yes | Yes |
| OFPT_PORT_MOD | No | No |
| OFPT_TABLE_MOD | No | No |
| OFPT_MULTIPART_REQUEST | Yes | Yes |
| OFPT_MULTIPART_REPLY | Yes | Yes |
| OFPT_BARRIER_REQUEST | Yes | Yes |
| OFPT_BARRIER_REPLY | Yes | Yes |
| OFPT_QUEUE_GET_CONFIG_REQUEST | No | No |
| OFPT_QUEUE_GET_CONFIG_REPLY | No | No |
| OFPT_ROLE_REQUEST | Yes | Yes |
| OFPT_ROLE_REPLY | Yes | Yes |
| OFPT_GET_ASYNC_REQUEST | Yes | Yes |
| OFPT_GET_ASYNC_REPLY | Yes | Yes |
| OFPT_SET_ASYNC | Yes | Yes |
| OFPT_METER_MOD | Yes | Yes |

## Multiple controllers

An OpenFlow switch may be connected to multiple controllers for reliability, allowing the switch to continue to operate in OpenFlow mode if a controller or controller connection fails. The controllers coordinate the management of the switch amongst themselves to help synchronize controller handoffs.

Each controller can have one of the following roles:

- Equal (OFPCR_ROLE_EQUAL): The controller has full access to the switch. It receives all the asynchronous messages from the switch and sends commands to modify the state of the switch (add or delete flows).
- Slave (OFPCR_ROLE_SLAVE): The controller has a read-only access to the switch. It does not receive the asynchronous messages (apart from port status). It does not execute commands that modify the state of the switch: **packet-out, flow-mod, group-mod**, or **port-mod**. The switch must reply with an OFPT_ERROR message, if it receives one of those commands from a Slave controller. Other controller-to-switch messages are processed normally.
- Master (OFPCR_ROLE_MASTER): The controller has full access to the switch as in the Equal role. When the controller changes its role to Master, the switch changes the other controller in the Master role to have the Slave role. The role change does not affect controllers with the Equal role.

Regardless of the intended use of multiple controller connections, the device allows all the controller connections to concurrently manage the flow table. That is, flow entries in the flow table are not identified as belonging to any specific controller connection. In an active-standby controller deployment, controllers themselves must coordinate their actions and active-standby states. The device responds to all connected controllers without distinction.

The device supports the active controller connection (also called mode). The device initiates the TCP connection to a given OpenFlow Controller address. A counter Generation ID is assigned by the controller each time the mastership view changes. For role changing to Master or Slave, the switch validates Generation ID to check for stale messages.

## *Asynchronous configuration*

Asynchronous messages may need to be sent to multiple controllers. An asynchronous message is duplicated for each eligible OpenFlow channel, and each message is sent when the respective controller connection allows it.

A controller can also control which types of switch asynchronous messages are sent over its OpenFlow channel. This is done using an asynchronous configuration message that has the filter setting for all the messages.

Each role for every controller may have its own set of asynchronous message setting. A controller in the Master role can selectively disable notifications, and a controller in the Slave role can enable notifications it wants to monitor.

Each controller configuration block for active connection maintains its own asynchronous configuration setting for every role. The default initial configuration is shown in the following table.

**TABLE 8** Action for asynchronous configuration

| Messages | Bit field | Master or Equal role | Slave role |
|---|---|---|---|
| Packet-in reasons | No_match | Enable | Disable |
| | Action | Enable | Disable |
| | Invalid_TTL | Disable | Disable |
| Port status reasons | Add | Enable | Enable |
| | Delete | Enable | Enable |
| | Modify | Enable | Enable |
| Flow removed reasons | Idle_timeout | Disable | Disable |
| | Hard_timeout | Disable | Disable |
| | Delete | Enable | Disable |
| | Group_delete | Enable | Disable |

NOTE
The asynchronous messages, Action, Invalid_TTL, Idle_timeout, and Hard_timeout are not supported by devices. Controllers can set these bits in the filter setting and the device can accept the bits, but the messages are not sent out by the device.

# OpenFlow configuration

You can enable OpenFlow on an interface with Layer23 flows in order to support Layer 2 and Layer 3 flows on that interface. Layer23 flows support the OpenFlow hybrid port mode also. Configured with Layer23, the controller can configure flows with Layer 2 and Layer 3 parameters together. A flow can contain the following fields: Ingress port, Destination MAC address, Source MAC address, Ether type, VLAN ID, P-bits, Source IP address, Destination IP address, IP protocol, and IP DSCP.

By default, OpenFlow is disabled on devices. You must first enable OpenFlow on the device before you can configure the parameters on the device.

## Enabling OpenFlow on devices

After you enable OpenFlow on the device, you can enable OpenFlow on specific interfaces and configure additional OpenFlow parameters.

To enable OpenFlow, enter the following command:

```
device(config)# openflow enable ofv13
```

The **ofv13** keyword specifies the OpenFlow protocol version supported.

Use the **no openflow enable ofv13** command to disable OpenFlow on the device.

NOTE
You must disable OpenFlow on all interfaces individually before you can disable OpenFlow globally on the device.

## Enabling OpenFlow on a specified interface

After you have enabled OpenFlow on the device, you can enable OpenFlow on specific interfaces.

NOTE
You can enable OpenFlow on an interface only after you have enabled OpenFlow globally on the device. In addition, you must use individual CLI commands to enable OpenFlow on each interface. You cannot specify a range of ports when enabling OpenFlow.

NOTE
Configuration of an OpenFlow hybrid port is not supported, if the port is already configured as a member of an MCT VLAN.

To enable OpenFlow on different interfaces, enter the following commands:

```
device(config)#interface Ethernet 3/1
device(conf-if-eth-3/1)#openflow enable layer2
device(config)#interface Ethernet 3/2
device(conf-if-eth-3/2)#openflow enable layer3
device(config)#interface Ethernet 3/3
device(conf-if-eth-3/3)#openflow enable layer23
```

If you enable Layer 3 matching mode on the specified interface, only Layer 3 matching fields are supported on that interface.

Use the **no openflow enable** command to disable OpenFlow on the interface.

# Configuring the OpenFlow Controller

To configure the OpenFlow controller, use the following steps.

By default no controller connection is present. The device supports up to three controller connections.

Set the IP address of the Controller.

1. From privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter **openflow enable** command to globally enable OpenFlow.

   ```
   device(config)# openflow enable ofv13
   ```

   The ofv13 option represents OpenFlow version 1.3.0.
3. Enter **openflow controller** command to name the controller and assign an IP address.

   ```
   device(config)# openflow controller A1 ip-address 10.25.128.185 no-ssl
   ```

   The OpenFlow controller is created to use TCP connectivity without SSL. By default, a controller role is Equal. The controller may change its role using OFPT_ROLL_REQUEST message.
4. Verify the OpenFlow controller configuration.

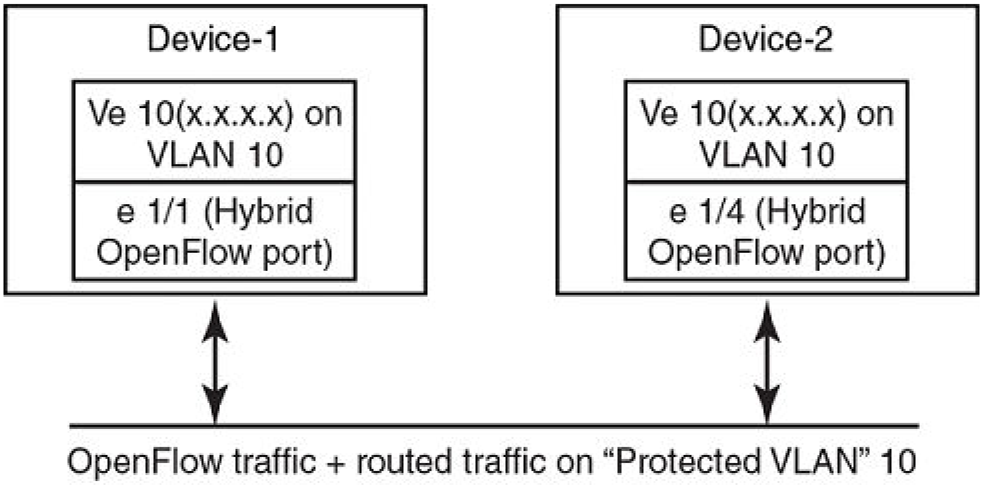
   ```
   device(config)# openflow controller A1 ip-address 10.25.128.185 no-ssl port 9000
   device(config)# openflow controller A2 ip-address 10.25.128.185 no-ssl
   device(config)# no openflow controller A2
   ```

# OpenFlow hybrid port mode

OpenFlow hybrid-enabled ports support both OpenFlow traffic forwarding and normal routing traffic forwarding. OpenFlow hybrid-enabled ports support "protected VLANs" and "unprotected VLANs". Protected VLANs are not subject to defined OpenFlow flows on the OpenFlow hybrid-enabled ports. OpenFlow flows on a hybrid-enabled port will not match any traffic on protected VLANs. Unprotected VLANs are subject to defined OpenFlow flows on the OpenFlow hybrid-enabled port. OpenFlow flows on a hybrid-enabled port are allowed to match on the traffic of unprotected VLANs.

The following figure shows a topology in which port 1/1 on Device-1 and port 1/4 on Device-2 are hybrid-enabled OpenFlow ports with VLAN 10 as a configured protected VLAN. By configuring a virtual Ethernet (VE) interface on a protected VLAN 10 and assigning an address to route the traffic of the nodes, you are able to send protected VLAN traffic between the nodes and route the traffic as per the VE interface. Traffic flowing on other VEs created on top of other VLANs (the unprotected VLANs ) is treated as unprotected VLAN traffic and is subject to OpenFlow rules lookup. OpenFlow traffic can be forwarded through this port.

**FIGURE 2** OpenFlow hybrid port mode topology


Device-1
Ve 10(x.x.x.x) on VLAN 10
e 1/1 (Hybrid OpenFlow port)
Device-2
Ve 10(x.x.x.x) on VLAN 10
e 1/4 (Hybrid OpenFlow port)
OpenFlow traffic + routed traffic on "Protected VLAN" 10

# OpenFlow hybrid port mode operation

Ingress traffic on VLAN 10 on hybrid port 1/1 is processed for IPv4 and IPv6 unicast routing. Traffic on other VLANs is processed against OpenFlow flows on port 1/1 and switched accordingly. A preconfigured number of protected VLANs can be supported for normal routing.

# Configuring OpenFlow hybrid port mode

1. Enable OpenFlow at the global configuration level.
2. Configure OpenFlow controller configurations.

   NOTE
   The port needs to be configured in switchport trunk mode, before configuring as an OpenFlow hybrid port.

3. Configure the maximum OpenFlow unprotected VLAN entries. (The default is 0.)
4. Configure protected VLANs on the port. A maximum of 40 protected VLANs can be configured on an OpenFlow port.
5. Enable OpenFlow hybrid port mode on the desired interfaces.
6. Configure a VE for the interface by specifying the protected VLAN and add routing entries.

# Capabilities, limitations and prerequisites for hybrid ports

The following are current capabilities, limitations and prerequisites of OpenFlow hybrid port mode.

## *Capabilities and prerequisites*

- The system supports maximum of 40 protected VLANs per port.
- Up to 2K protected VLANs per system are supported.
- OpenFlow v1.3 protocol is supported.

- Layer 2 protocols and VPLS forwarding are supported on ports in hybrid-enabled ports.
- IPv4 and IPv6 unicast routing are supported on OpenFlow protected and unprotected VLANs.
- Packets tagged with a protected VLAN ID are forwarded by IPv4 and IPv6 unicast routing, if IPv4 or IPv6 routing is configured on that VLAN. If IPv4 or IPv6 routing is not configured on that VLAN (unconfigured VLAN), such packets are dropped.
- Packets tagged with an unprotected VLAN ID are subject first to OpenFlow flows. If there is a match on an OpenFlow flow, the packet is forwarded according to the flow actions. No further IPv4 or IPv6 routing is supported for packets that are forwarded by OpenFlow flows. If there is no match on any OpenFlow flow, the packet is forwarded by IPv4 or IPv6 unicast routing, if IPv4 or IPv6 routing is configured on the VLAN. If IPv4 or IPv6 routing is not configured on the VLAN, that packet is either dropped or sent to the controller, per the OpenFlow configuration.
- The BGP, OSPF, and IS-IS protocols are supported on protected VLANs.
- Layer 2 protocols like MSTP, SSTP, or PVSTP are supported.
- Port level sFlow sampling is supported.
- If a flow from the controller is added for a protected VLAN, or VLAN configured as protected on the port, such flow is rejected.
- If a flow belongs to a VLAN, then that VLAN is not allowed to configure as protected VLAN.

### *Limitations*

- OpenFlow hybrid port cannot be an untagged member of any VLAN or VPLS VLAN except the default VLAN.
- OpenFlow hybrid cannot be enabled on a port configured as switchport mode access.
- Protected VLAN traffic with no matching MAC or IP route entries is dropped.
- A port can be enabled for OpenFlow hybrid port mode only if the port is untagged in the default VLAN.
- Policy-based routing (PBR) is not supported.
- Hybrid port cannot be configured as Layer 3 port purely.
- Link level Layer 2 protocols (UDLD, 802.1x, loop detect, link-OAM, LLDP, LACP) are not supported on OpenFlow interfaces.
- LAG interface cannot be a hybrid port, and a hybrid port cannot be added to a LAG interface.
- Inbound normal ACL configuration is not supported on the port in hybrid port mode.
- Wild carded VLAN flows existing in the system does not affect the individual VLANs getting configured as protected.

## Enabling OpenFlow hybrid port mode

Use the **openflow enable hybrid-mode** command to enable OpenFlow hybrid port mode on the port for different interfaces. The **no openflow enable hybrid-mode** command disables the OpenFlow hybrid port mode on the port and the port becomes a normal port. For example, to enable OpenFlow hybrid mode on Layer 2, use the following command.

```
device(config-if-eth-3/1)# openflow enable layer2 hybrid-mode
```

To enable OpenFlow hybrid mode on Layer 23, use the following command.

```
device(config-if-eth-3/1)# openflow enable layer23 hybrid-mode
```

## Configuring OpenFlow hybrid port on an interface

1. Enable OpenFlow at the global configuration level.
2. Configure OpenFlow controller configurations.
3. Configure the switchport on the interface.

4. Configure the switchport trunk mode on the interface.

The following example configures an OpenFlow hybrid port on an interface.

```
device(config)# interface Ethernet 3/1
device(conf-if-eth-3/1)# switchport
device(conf-if-eth-3/1)# switchport mode trunk
device(conf-if-eth-3/1)# openflow enable layer23 hybrid-mode
```

## Adding or deleting protected VLANs

Use the **openflow protected-vlans** command to add or delete protected VLANs on an OpenFlow hybrid port mode interface. The **no openflow protected-vlans** command deletes the configured protected VLANs from the hybrid-enabled port.

```
device(config)# interface Ethernet 3/1
device(config-if-eth-3/1)# openflow protected-vlans add 101-105, 110
device(config-if-eth-3/1)# openflow protected-vlans remove 102-104, 110
```

NOTE
1. Both range or individual VLAN are supported for addition and removal.
2. The **openflow protected-vlans** can be preconfigured, even when hybrid mode is not enabled on an interface though, but global **openflow enable** is the requirement on the switch.
3. The port does not be in trunk mode before pre-provisioning a protected VLAN.

# QinQ

You can push, pop or set VLAN tags in the outgoing packets of an OpenFlow flow with QinQ support. The ingress packet can be untagged or tagged. You can use QinQ to transport multiple customer segments or VLANs across Layer 2 infrastructures.

A OpenFlow flow matches on VLAN and it does one of the following.

- Push VLAN
- Pop VLAN
- Set field (VLAN ID or VLAN-PCP)

A packet is identified as tagged if the tag-type is 0x8100, packet with any other tag type is considered as untagged. For example, packet received with tag-type 0x9100 is an untagged packet.

NOTE
QinQ supports only one tag modification.

## QinQ action

The following table illustrates the behavior of push or set field VLAN ID for different tags.

TABLE 9 OpenFlow QinQ actions

| OpenFlow action | Input traffic | |
|---|---|---|
| | Untagged packet | Tagged packet |
| Push VLAN | New VLAN header is added with the given tag-type and VLAN ID. | New VLAN header is added with the given tag-type and VLAN ID. |
| Set VLAN | New VLAN header is added with default tag-type 0x8100 and set field VLAN ID. | Outermost VLAN header is modified. |

**TABLE 9** OpenFlow QinQ actions (continued)

| OpenFlow action | Input traffic | |
|---|---|---|
| | Untagged packet | Tagged packet |
| Pop VLAN | Pop action is ignored. | Outermost VLAN header is removed. |

OpenFlow flow is rejected, if there is mismatch in tag-type of port and the push action.

The following table illustrates the behavior of set field VLAN-PCP along with other set or push VLAN action.

**TABLE 10** OpenFlow QinQ actions for set field VLAN-PCP

| Incoming traffic | Redirect action | VLAN action | | |
|---|---|---|---|---|
| | | Push VLAN + Set VLAN-PCP | Set VLAN + Set VLAN-PCP | Set VLAN-PCP |
| Untagged packet | One output port | VLAN pushed along with new VLAN-PCP. | Both VLAN and VLAN-PCP are added as the outer VLAN header. | Not applicable |
| | Multiple output port | Not supported. | VLAN header is added without VLAN-PCP. | Not applicable |
| Tagged packet | One output port | VLAN pushed along with new VLAN-PCP. | Both VLAN and VLAN-PCP are modified. | VLAN-PCP is modified. |
| | Multiple output port | Not supported. | Both VLAN and VLAN-PCP are modified. | VLAN-PCP is modified. |

NOTE
OpenFlow flow is rejected, when it is not supported. OpenFlow action is silently ignored, when it is not applicable.

# Group table

The group table introduces the ability to add support for port group abstraction for multi-pathing. This enables OpenFlow to represent a set of ports as a single entity for forwarding packets.

The group table supports the following group types:

- All: Executes all the buckets in the group; mostly used for flooding and multicasting.
- Indirect: Executes one defined bucket in the group. The action taken by this group type is sending packets to the next hop.
- Select: Executes one bucket in the group. The action bucket is chosen by a switch-defined algorithm, such as round robin or hashing (for example, load sharing).
- Fast failover: Executes the first live bucket, used in cases such as redundancy.

A group table consists of group entries. The counters in the following table are available in a group entry.

**TABLE 11** Group entry counters

| Counter | Description |
|---|---|
| Group Identifier | A 32-bit unsigned integer uniquely identifying the group |
| Group type | Determines group semantics |
| Counter | Number of packets processed by a group |
| Action bucket | Ordered list of action buckets, where each action bucket contains a set of actions to execute and associated parameters |

# Group messages

The following table describes the processing of group messages.

**TABLE 12** Group messages

| Group message type | Entry exists | Entry does not exist | Notes |
|---|---|---|---|
| Add (OFPGC_ADD) | Deny Add. Return error message to controller | Add is processed | Subject to constraints below |
| Mod (OFPGC_MODIFY) | Group parameters and action buckets are updated | Deny Mod. Return error message to controller | Update or modify is implemented, as delete followed by add in the driver. |
| Del (OFPGC_DELETE) | Group entry is deleted. Flows which are associated with this group are also removed. | No Error. Message ignored | If a Del comes in, that has flows associated with it, then delete those flows from the system. |

## *Error conditions and messages*

This table lists the error conditions and the error opcodes sent to the controller. The error type is always OFPET_GROUP_MOD_FAILED.

**TABLE 13** Group messages

| Error condition | Opcode |
|---|---|
| Adding group, if group already exists | OFPGMFC_GROUP_EXISTS |
| When group allocation exceeds memory or system limit | OFPGMFC_OUT_OF_GROUPS |
| Group type is not supported | OFPGMFC_BAD_TYPE |
| In case of group modification or deletion, if group does not exist | OFPGMFC_UNKNOWN_GROUP |
| Number of buckets in a group is greater than 8 in all group types except Indirect | OFPGMFC_OUT_OF_BUCKETS |
| Number of output ports in a bucket is greater than 1 | OFPGMFC_BAD_BUCKET |
| (For all group types) Not an output port action or set field VLAN_VID or OFPAT_PUSH_VLAN or OFPAT_POP_VLAN | OFPGMFC_BAD_BUCKET |
| Modify group for Select type to non Select type and vice versa. | OFPGMFC_EPERM |
| Fast Failover: The watch port is absent | OFPGMFC_BAD_WATCH |
| Fast Failover: The watch group is present | OFPGMFC_BAD_BUCKET |

# Displaying groups for the OpenFlow ports

To show the number of groups for the OpenFlow ports or for given group ID.

1. From privileged EXEC mode, enter global configuration mode.

   ```
   device # show openflow group
   ```

2. To show groups for a specific group ID, enter the following command.

   ```
   device # show openflow group 12
   ```

The following example shows the output of the **show openflow group** command.

```
device # show openflow group
Max number of total groups                             : 512
Max number of buckets per group                        : 8

TOTAL number of groups(Type:ALL) in the system         : 1
TOTAL number of groups(Type:SELECT) in the system      : 0
TOTAL number of groups(Type:Indirect)in the system     : 0
TOTAL number of groups(Type:Fast Failover)in the system : 0

TOTAL number of groups in the system                   : 1

Group id 1
  Transaction id        4043243760 (0xf0ff00f0)
  Type                  ALL
  Packet Count          0
  Byte   Count          0
  Flow   Count          0
  Number of buckets     2

  bucket #1
    Weight             1
    out port: Eth 3/25

  bucket #2
    Weight             2
    out port: Eth 3/26
----
```

# Metering

Per-flow metering measures and controls the rate of packets for each flow entry. Per-flow meters enable OpenFlow to implement simple QoS operations, such as rate-limiting, and can be combined with per-port queues to implement complex QoS frameworks, such as DiffServ.

Meters are attached directly to flow entries. Each meter can have one or more meter bands. Each meter band specifies the rate of the band applies and the way packets are processed (DROP or DIFFSERV). OpenFlow metering operation is similar to ingress rate limiting in a QoS operation.

A meter table consists of meter entries. The counters in the following table are available in the meter entry.

**TABLE 14** Meter entry

| Counter | Description |
|---|---|
| Meter Identifier | A 32-bit unsigned integer uniquely identifying the meter |
| Meter band | Each meter band specifies the rate of the band and the way to process the packet. Rate and burst size are based on the line rate of the data traffic in contrast to the information rate. |
| Counter | Number of packets processed by a meter |

Packets are processed by a single meter band based on the current measured meter rate. The meter applies the meter band with the highest configured rate that is lower than the current measured rate. If the current rate is lower than any specified meter band rate, no meter band is applied.

**TABLE 15** Meter band supported on devices

| Meter bands | Supported |
|---|---|
| DROP | Yes |

Each band type contains the following meter configuration parameters from the controller:

- Rate value in kbps
- Rate value in packets per second
- Burst size
- Statistics collection

The metering system supports the features in the following table.

**TABLE 16** Metering capabilities supported for metering features

| Feature | SLX 9850-4 | SLX 9850-8 |
|---|---|---|
| Max meter | 1K | 1K |
| Band types (bitmap) | DROP | DROP |
| Capabilities (bitmap) | KBPS, BURST | KBPS, BURST |
| Maximum color value | 2 (RED, GREEN) | 2 (RED, GREEN) |

## Limitations

- Minimum burst size for DROP band is 10 kbits.
- Maximum burst size supported in hardware is 33292 kbits.

## Meter messages

The following table describes the processing of the meter messages.

**TABLE 17** Meter messages

| Meter message type | Entry exists | Entry does not exist | Notes |
|---|---|---|---|
| Add (OFPMC_ADD) | Deny Add. Return error message to controller. | Add is processed. | Deny Add for the lack of memory or internal error or due to restrictions (hardware or otherwise) limiting the number of bands. Return error message to controller as per standard. |
| Mod (OFPMC_MODIFY) | Meter parameters and bands are updated. New bands replace the existing bands. | Deny Mod. Return error message to controller. | Meter Mod updates meter parameters and new bands replacing the existing bands. Forward layers apply these changes instantly or make before break mechanism. |
| Del (OFPMC_DELETE) | Meter entry and bands removed. Flows associated with this meter are also removed. | No Error. Message ignored. | Only the meter identifier is specified for the delete request. |

# Show commands for OpenFlow

Show commands for OpenFlow are included in the following table and described in detail in the *Extreme SLX-OS Command Reference*.

**TABLE 18** Show commands for OpenFlow

| Command | Description |
|---|---|
| **show openflow** | Shows all the OpenFlow configuration. |
| **show openflow flow** | Shows all the flows configured in the system flow table. |
| **show openflow controller** | Shows the status of all the controllers. |
| **show openflow interface** | Displays the ports with OpenFlow. |
| **show openflow resources** | OpenFlow usage of the resources. |
| **show openflow queues** | Shows the queue entries for the interface. |
| **show openflow group** | Shows all the groups in a flow. |
| **show openflow meter** | Shows all the meters in a flow. |

# Clear commands for OpenFlow

Clear commands for OpenFlow are included in the following table and described in detail in the *Extreme SLX -OS Command Reference*.

When an individual OpenFlow rule or all flows in the flow table need to be deleted, you can use the **clear openflow** command. Use this command to delete a single OpenFlow rule based on a Flow ID or delete all flows/groups/meters configured in the system.

You can clear the flow statistics for all flows or for a specified flow.

**TABLE 19** Clear commands for OpenFlow

| Command | Description |
|---|---|
| **clear openflow** | Clears a single OpenFlow rule based on a Flow ID or deletes all flows/ groups/meters configured in the system. |
| **clear statistics openflow** | Clears statistics for the controller, flow, all groups, or all meters. |