Configuring Open Flow Agent

OpenFlow is a specification from the Open Networking Foundation (ONF) that defines a flowbased forwarding infrastructure (L2-L4 Ethernet switch model) and a standardized application programmatic interface (protocol definition) to learn capabilities, add and remove flow control entries and request statistics. OpenFlow allows a controller to direct the forwarding functions of a switch through a secure channel.

This module has details about the Open Flow Agent, relevant concepts and configurations.

Table 1: Feature History for Implementing OFACisco IOS XR Software

Release	Modification
Release 5.1.2	This feature was introduced.
Release 5.3.4	OnePK support was discontinued.

OpenFlow

Openflow is an open standard to communicate between controllers, which are running applications and network elements (such as, routers and switches).

For details regarding OpenFlow, please refer the OpenFlow chapter in the *System Management Configuration Guide for Cisco ASR 9000 Series Routers* .

An overview of OFA

OpenFlow is a specification from the Open Networking Foundation (ONF) that defines a flowbased forwarding infrastructure (L2-L4 Ethernet switch model) and a standardized application programmatic interface (protocol definition) to learn capabilities, add and remove flow control entries and request statistics. OpenFlow allows a controller to direct the forwarding functions of a switch through a secure channel. Local device configuration is out of scope of the OpenFlow protocol. OpenFlow essentially provides a forwarding instruction set, allowing applications to directly program any-to-any routing and switching, with header field rewrite. New matches and actions can be applied to packets in arbitrary unconstrained fashion, allowing routing and switching on the new criteria. Routers and switches embed the fast packet forwarding and the high level routing decisions together into their software on the same device. With only a few exceptions based on user configuration, all routing and switching decisions are made by the built-in protocols and control plane logic that reside on the switch.

Prerequisites for OpenFlow Agent

The following prerequisites are required to use the OpenFlow agent on the platforms supporting IOS-XR:

- Special build of the Release 5.1.x software that has the OpenFlow functionality is required.
- The Enhanced Ethernet line card for the Cisco ASR 9000 Series Router is required for the OpenFlow agent feature.
- Any controller with version 1.1 or 1.3 is required (example, POX, ODL).
- The asr9k-k9sec Package Installation Envelope (PIE) must be present. The asr9k-mpls PIE is required for support on MPLS core (such as, PWHE).

Restrictions for OpenFlow Agent

- Same interface cannot be added to more than one logical open flow switch.
- No support for output as an action for layer3 openflow logical switch (such as pipeline131, 132).
- Only layer 3 interface support for netflow sampling statistics.

Advantages

The advantages with Open Flow Agent are:

- increases network scalability
- reduces network complexity
- allows greater application control

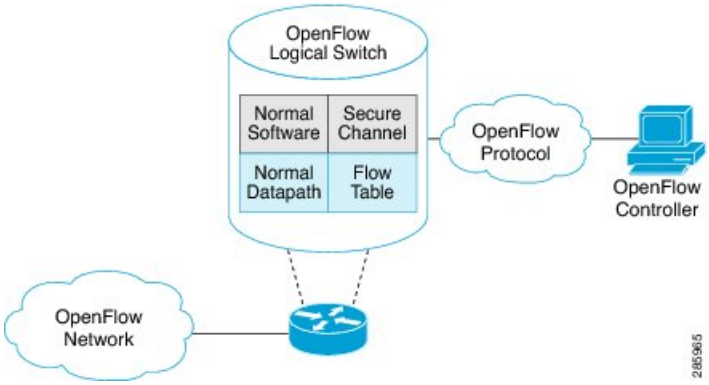
- enables customer-feature-independence

About OpenFlow

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.

In the Cisco OpenFlow scheme, the physical switch is divided into multiple logical switches by using the CLI to configure the connection to the controller for each logical switch and enable interfaces for each logical switch. The Openflow Agent software manages these logical switches.

The following figure shows the Cisco implementation of the OpenFlow network.



Openflow Mode for ASR9000

Openflow for the Cisco ASR 9000 Series router functions in the Integrated Hybrid mode. In this mode, both Openflow and normal switching and routing (for layer 3) operations such as L2 ethernet switching, L3 routing, etc are supported. Packets processed as the Openflow forwarding path can be processed as a normal forwarding path.

OpenFlow Table Types

An OpenFlow flow table consists of a set of flows. Each flow contains a set of matches and actions. A table has a set of capabilities in terms of supported matches and actions. Just like a policy-map, a table can be applied to a set of targets but only in the ingress direction. Hence, OpenFlow matches and actions are applied to the incoming traffic only.

Note A set of ordered tables is referred to as a pipeline. A pipeline may contain one or more ordered tables. An OpenFlow pipeline of an OpenFlow switch on ASR9K supports only one flow table.

Table 2: OpenFlow Table Types

Table Type	Pipeline	Supported Interfaces	Description
L2	129	Bridge-domain, Gigabit ethernet, Bundle, Bundle-subinterfaces, PWHE-subinterfaces	• Supports L2 header matches. • Supports L2 actions. • Can be applied to the ingress L2 interfaces.
L2_L3	130	Bridge-domain, Gigabit ethernet, Bundle, Bundle-subinterfaces, PWHE-subinterfaces	• Supports L2 and L3 (IPv4/IPv6) header matches. • Supports L2 actions. • Can be applied to the ingress L2 interfaces.
L3_V4	131	VRF and global interfaces, BVI (ipv4 only), Bridge-domain, Gigabit ethernet, Bundle, Bundle-subinterfaces	• Supports L3 (IPv4) header matches. • Supports L3 (IPv4) actions. • Can be applied to the ingress L3 interfaces.
L3_DS	132	VRF and global interfaces, BVI, Bridge-domain, Gigabit ethernet, Bundle, Bundle-subinterfaces	• Supports L2 and L3 (IPv4/IPv6) header matches. • Supports L3 (IPv4/IPv6) actions. • Can be applied to the ingress L3 interfaces.

- L2 Table--Supports L2 header matches and has L2 actions only. This table type can be applied to the ingress of an L2 interface.
- L2_L3 Table--Supports L2 and L3 header matches and has L2 actions only. Match parameters can be IPv4 or IPv6 type. This table type can be applied to the ingress of an L2 interface.
- L3_V4 Table--Supports L3 IPv4 header matches and has L3 actions only. This table type can be applied to the ingress of L3 interfaces.
- L3_DS(Dual Stack) Table--Supports L2 and L3 IPv4 and IPv6 (Dual Stack) matches and has L3 actions only. This table type can be applied to the ingress of L3 interfaces.

OpenFlow Agent Packet In and Out Feature

The Packet In and Out feature allows a flow to be programmed by the OpenFlow Agent logical switch so that packets are sent to the Controller. The special output port: **OFP_CONTROLLER** is specified for the flow action.

The Packet In and Out feature enables support for the OpenFlow output-to-port action. The output action tells the OpenFlow Agent to send all packets matching the flow to a specific port.

OpenFlow Agent with NetFlow Collection and Analytics

Applications can be provided with on-demand analytics by using the OpenFlow protocol with NetFlow. NetFlow provides statistics on packets flowing through the router, and is the standard for acquiring IP operational data from IP networks.

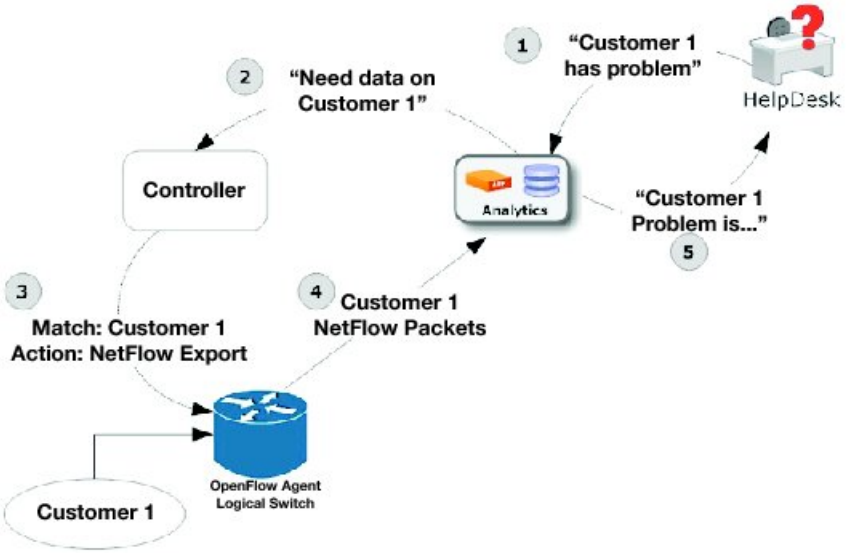
The following NetFlow maps must be configured:

- Flow Exporter Map—Specifies the destination IP address of the NetFlow collector where the NetFlow Version 9 packets are sent.
- Flow Monitor Map—Specifies the profile of the NetFlow producer, including the timeout values of active and inactive timers, size of the NetfFow cache and the exporter to be used.
- Sampler Map—Specifies how often Network Processor (NPU) needs to sample incoming and outgoing packets and create flow-packets to punt to the Line Card (LC) Central Processing Unit (CPU).

The following parameters must be specified on the OpenFlow Agent logical switch:

- Interface associated with the OpenFlow Agent logical switch that is enabled for NetFlow.
- Flow Monitor Map
- Sampler Map
- Controller IP address

Figure 1: OpenFlow Agent and NetFlow collection and analytics workflow



1. The help desk application tells the analytics application that Customer 1 has a problem.
2. The analytics application determines that it requires more information and requests more network data about Customer 1 from the Controller.
3. The Controller instructs the OpenFlow logical switch on the router to look for Customer 1 packets and generate and export NetFlow data based on Customer 1 packet flows.

4. The OpenFlow Agent logical switch exports NetFlow packets to the analytics application where they are processed.
5. The analytics application informs the help desk application of the problem.

OFA on Cisco Routers and Switches

OpenFlow SDN Applications expect network elements to speak standard OpenFlow protocol and to implement standard OpenFlow switch model. The OpenFlow Agent as a local process provides:

- OF protocol stack
- OF switch model derived from disparate Cisco software and hardware
- Version, model and feature negotiation
- Local aggregation of state and statistics
- Native dedicated CLI and troubleshooting
- High Availability

Functional Components

OpenFlow supports the configuration of multiple controllers for a logical switch. The Openflow agent can connect to a single controller or up to 8 controllers. It creates connections to all configured controllers to provide the controllers access to the OpenFlow logical switch flow tables and interfaces. It will receive flow entries from the controllers and report interface and flow status and statistics to the controllers.

The set nexthop action for layer 3 matches is implemented through a Cisco extension to the OpenFlow (1.0 and 1.3) protocol.

OFA on ASR 9000 series routers

The OpenFlow Agent supports multiple logical switch instances on ASR9K platform, with each logical switch managing a set of physical/logical interfaces, an L2 bridge domain or a VRF. Each logical switch may have one openflow connection to a single controller, or multiple connects for reliability, each to a different controller . The openflow connection to the controller uses standard TLS or plain TCP.

When the logical switch initialises a connection to the configured controller, the signaling version for the agent-controller connection is negotiated based on the bitmap version supported on both- agent and controller sides. When a logical switch starts up for the first time or at the time a logical switch loses contact with all controllers, it operates in either fail-secure mode (with default-set rule) or fail-standalone mode depending on the CLI of fail-standalone (on or off). The default for configuration is in the fail-secure mode.

OpenFlow Matches

Matches are supported on ingress port and various packet headers depending upon the packet type. Flows can have priorities. Hence, the highest priority flow entry that matches the packet gets selected.

Following table shows the list of matches supported on ASR9K for various table types:

OpenFlow Matches		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OXM Flow match field type for OpenFlow basic class	**Description**	**L2 only**	**L2_L3**	**L3_V4**	**L3_DS**
OFPXMT_OFB_IN_PORT	Switch input port	Yes	Yes	Yes	Yes
OFPXMT_OFB_IN_PHY_PORT	Switch physical port	No	No	No	No
OFPXMT_OFB_METADATA	Metadata passed between tables	No	No	No	No
OFPXMT_OFB_ETH_DST	Ethernet destination address	Yes	Yes	No	Yes
OFPXMT_OFB_ETH_SRC	Ethernet source address	Yes	Yes	No	Yes
OFPXMT_OFB_ETH_TYPE	Ethernet frame type	Yes	Yes	No	Yes
OFPXMT_OFB_VLAN_VID	VLAN ID	Yes	Yes	No	Yes
OFPXMT_OFB_VLAN_PCP	VLAN priority	Yes	Yes	No	Yes
OFPXMT_OFB_IP_DSCP	IP DSCP (6 bits in ToS field)	No	Yes	Yes	Yes
OFPXMT_OFB_IP_ECN	IP ECN (2 bits in ToS field)	No	No	No	No
OFPXMT_OFB_IP_PROTO	IP protocol	No	Yes	Yes	Yes
OFPXMT_OFB_IPV4_SRC	IPv4 source address	No	Yes	Yes	Yes
OFPXMT_OFB_IPV4_DST	IPv4 destination address	No	Yes	Yes	Yes
OFPXMT_OFB_TCP_SRC	TCP source port	No	Yes	Yes	Yes
OFPXMT_OFB_TCP_DST	TCP destination port	No	Yes	Yes	Yes

OpenFlow Matches		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OFPXMT_OFB_UDP_SRC	UDP source port	No	Yes	Yes	Yes
OFPXMT_OFB_UDP_DST	UDP destination port	No	Yes	Yes	Yes
OFPXMT_OFB_SCTP_SRC	SCTP source port	No	Yes	Yes	Yes
OFPXMT_OFB_SCTP_DST	SCTP destination port	No	No	No	No
OFPXMT_OFB_ICMPV4_TYPE	ICMP type	No	No	No	No
OFPXMT_OFB_ICMPV4_CODE	ICMP code	No	No	No	No
OFPXMT_OFB_ARP_OP	ARP opcode	No	No	No	No
OFPXMT_OFB_ARP_SPA	ARP source IPv4 address	No	No	No	No
OFPXMT_OFB_ARP_TPA	ARP target IPv4 address	No	No	No	No
OFPXMT_OFB_ARP_SHA	ARP source hardware address	No	No	No	No
OFPXMT_OFB_ARP_THA	ARP target hardware address	No	No	No	No
OFPXMT_OFB_IPV6_SRC	IPv6 source address	No	Yes	No	Yes
OFPXMT_OFB_IPV6_DST	IPv6 destination address	No	Yes	No	Yes
OFPXMT_OFB_IPV6_FLABEL	IPv6 Flow Label	No	No	No	No
OFPXMT_OFB_ICMPV6_TYPE	ICMPv6 type	No	No	No	No
OFPXMT_OFB_ICMPV6_CODE	ICMPv6 code	No	No	No	No
OFPXMT_OFB_IPV6_ND_TARGET	Target address for ND	No	No	No	No
OFPXMT_OFB_IPV6_ND_SLL	Source link-layer for ND	No	No	No	No

OpenFlow Matches		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OFPXMT_OFB_IPV6_ND_TLL	Target link-layer for ND	No	No	No	No
OFPXMT_OFB_MPLS_LABEL	MPLS label	No	No	No	Yes
OFPXMT_OFB_MPLS_TC	MPLS TC	No	No	No	No
OFPXMT_OFP_MPLS_BOS	MPLS BoS bit	No	No	No	Yes
OFPXMT_OFB_PBB_ISID	PBB I-SID	No	No	No	No
OFPXMT_OFB_TUNNEL_ID	Logical Port Metadata	No	No	No	No
OFPXMT_OFB_IPV6_EXTHDR	IPv6 Extension Header pseudo-field	No	No	No	No

OpenFlow Actions

Packet forwarding and packet modification types of actions are supported. The lists of actions are always immediately applied to the packet.

Note

- Only “Apply-actions” instruction (OFPIT_APPLY_ACTIONS) of OpenFlow 1.3 is supported.
- Pipeline processing instructions that allow packets to be sent to subsequent tables for further processing are not supported in this release.
- Group tables and Meter tables are not supported.

Following table shows the list of action types supported on ASR9K for various table types.

OpenFlow Actions		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OXM Flow action field type for OpenFlow basic class	**Description**	**L2 only**	**L2_L3**	**L3_V4**	**L3_DS**
OFPAT_OUTPUT	Output to switch port.	Yes	Yes	No	No
OFPAT_COPY_TTL_OUT	Copy TTL "outwards"	No	No	No	No

OpenFlow Actions		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OFPAT_COPY_TTL_IN	Copy TTL "inwards"	No	No	No	No
OFPAT_SET_MPLS_TTL	MPLS TTL	No	No	No	No
OFPAT_DEC_MPLS_TTL	Decrement MPLS TTL	No	No	No	No
OFPAT_PUSH_VLAN	Push a new VLAN tag	Yes	Yes	No	No
OFPAT_POP_VLAN	Pop the outer VLAN tag	Yes	Yes	No	No
OFPAT_PUSH_MPLS	Push a new MPLS tag	No	No	No	No
OFPAT_POP_MPLS	Pop the outer MPLS tag	No	No	No	No
OFPAT_SET_QUEUE	Set queue id when outputting to a port	No	No	No	No
OFPAT_GROUP	Apply group	No	No	No	No
OFPAT_SET_NW_TTL	IP TTL	No	No	No	No
OFPAT_DEC_NW_TTL	Decrement IP TTL	No	No	No	No
OFPAT_SET_FIELD	Set a header field using OXM TLV format	Yes	Yes	Yes	Yes
OFPAT_PUSH_PBB	Push a new PBB service tag (I-TAG)	No	No	No	No
OFPAT_POP_PBB	Pop the outer PBB service tag	No	No	No	No

Cisco Extension Actions

The set ipv4 or set ipv6 nexthop actions are used to redirect an ipv4 or ipv6 packet to the specified nexthop address, instead of using the destination address in the packet. This provides ABF (ACL Based Forwarding) kind of functionality using OpenFlow. However, VRF support and nexthop tracking as supported by CLI based ABF feature is not supported in this release.

The set fcid (Forward Class ID) action can be used to support PBTS (Policy Based Tunnel Selection) functionality using OpenFlow.

Following table shows the list of actions added by Cisco to support some extra features on ASR9K.

Cisco proprietary actions		**OpenFlow Switch Types Supported on ASR9K**			
		Applied to L2 Bridge domain		**Applied to L3 or L3 VRF interface**	
OXM Flow match field type for OpenFlow basic class	**Description**	**L2 only**	**L2_L3**	**L3_V4**	**L3_DS**
Set Ipv4 Nexthop	Set ipv4 nexthop address	No	No	Yes	Yes
Set Ipv6 Nexthop	Set ipv6 nexthop address	No	No	No	Yes
Set Forward Class ID	Set forward class ID	No	No	Yes	Yes
Set VRF	Set forward ipv4/ipv6 packet based on VRF	No	No	Yes	Yes

Set Field Actions

This table lists the set field actions supported by the Cisco ASR 9000 series router:

OpenFlow Matches		**OpenFlow Switch Types Supported on ASR9K**			
		Applied to L2 Bridge domain		**Applied to L3 or L3 VRF interface**	
OXM Flow match field type for OpenFlow basic class	**Description**	**L2 only**	**L2_L3**	**L3_V4**	**L3_DS**
OFPXMT_OFB_ETH_DST	Ethernet destination address	Yes	Yes	No	No
OFPXMT_OFB_ETH_SRC	Ethernet source address	Yes	Yes	No	No
OFPXMT_OFB_ETH_TYPE	Ethernet frame type	No	No	No	No
OFPXMT_OFB_VLAN_VID	VLAN ID	Yes	Yes	No	No
OFPXMT_OFB_VLAN_PCP	VLAN priority	Yes	Yes	No	No

OpenFlow Matches		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OFPXMT_OFB_IP_DSCP	IP DSCP (6 bits in ToS field)	No	No	Yes	Yes
OFPXMT_OFB_IP_ECN	IP ECN (2 bits in ToS field)	No	No	No	No
OFPXMT_OFB_IP_PROTO	IP protocol	No	No	No	No
OFPXMT_OFB_IPV4_SRC	IPv4 source address	No	No	Yes	Yes
OFPXMT_OFB_IPV4_DST	IPv4 destination address	No	No	Yes	Yes
OFPXMT_OFB_TCP_SRC	TCP source port	No	No	Yes	Yes
OFPXMT_OFB_TCP_DST	TCP destination port	No	No	Yes	Yes
OFPXMT_OFB_UDP_SRC	UDP source port	No	No	Yes	Yes
OFPXMT_OFB_UDP_DST	UDP destination port	No	No	Yes	Yes
OFPXMT_OFB_SCTP_SRC	SCTP source port	No	No	No	No
OFPXMT_OFB_SCTP_DST	SCTP destination port	No	No	No	No
OFPXMT_OFB_ICMPV4_TYPE	ICMP type	No	No	No	No
OFPXMT_OFB_ICMPV4_CODE	ICMP code	No	No	No	No
OFPXMT_OFB_ARP_OP	ARP opcode	No	No	No	No
OFPXMT_OFB_ARP_SPA	ARP source IPv4 address	No	No	No	No
OFPXMT_OFB_ARP_TPA	ARP target IPv4 address	No	No	No	No
OFPXMT_OFB_ARP_SHA	ARP source hardware address	No	No	No	No

OpenFlow Matches		OpenFlow Switch Types Supported on ASR9K			
		Applied to L2 Bridge domain		Applied to L3 or L3 VRF interface	
OFPXMT_OFB_ARP_THA	ARP target hardware address	No	No	No	No
OFPXMT_OFB_IPV6_SRC	IPv6 source address	No	No	No	No
OFPXMT_OFB_IPV6_DST	IPv6 destination address	No	No	No	No
OFPXMT_OFB_IPV6_FLABEL	IPv6 Flow Label	No	No	No	No
OFPXMT_OFB_ICMPV6_TYPE	ICMPv6 type	No	No	No	No
OFPXMT_OFB_ICMPV6_CODE	ICMPv6 code	No	No	No	No
OFPXMT_OFB_IPV6_ND_TARGET	Target address for ND	No	No	No	No
OFPXMT_OFB_IPV6_ND_SLL	Source link-layer for ND	No	No	No	No
OFPXMT_OFB_IPV6_ND_TLL	Target link-layer for ND	No	No	No	No
OFPXMT_OFB_MPLS_LABEL	MPLS label	No	No	No	No
OFPXMT_OFB_MPLS_TC	MPLS TC	No	No	No	No
OFPXMT_OFP_MPLS_BOS	MPLS BoS bit	No	No	No	No
OFPXMT_OFB_PBB_ISID	PBB I-SID	No	No	No	No
OFPXMT_OFB_TUNNEL_ID	Logical Port Metadata	No	No	No	No
OFPXMT_OFB_IPV6_EXTHDR	IPv6 Extension Header pseudo-field	No	No	No	No

Configuring OneP for Openflow

SUMMARY STEPS

1. **configure**
2. **onep**

3. **datapath transport vpathudp sender-id** *number*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**onep** **Example:** `RP/0/RSP0/CPU0:router (config) # onep`	Enters the OneP configuration mode.
Step 3	**datapath transport vpathudp sender-id** *number* **Example:** `RP/0/RSP0/CPU0:router (config) # datapath transport vpathudp sender-id 1`	Configures the virtual-path udp transport datapath for the specified sender-id.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring a Layer 2 Logical Switch for the OpenFlow Agent

SUMMARY STEPS

1. **configure**
2. **openflow**
3. **switch** *switch -id* **pipeline** *pipeline-number*
4. **tls trust-point local** *local-tp-name* **remote** *remote-tp-name*
5. **bridge-group** *SDN-id* **bridge-domain** *switch-id*
6. **controller ipv4** *ip-address* **security [tls | none]**
7. **commit**
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**openflow** **Example:** `RP/0/RSP0/CPU0:router(config)# openflow`	Enters the openflow configuration mode.
Step 3	**switch** *switch -id* **pipeline** *pipeline-number* **Example:** `RP/0/RSP0/CPU0:router(config-openflow)# switch 1 pipeline 129`	Enters the logical switch configuration mode. For L2-only switch, the pipeline number is 129.
Step 4	**tls trust-point local** *local-tp-name* **remote** *remote-tp-name* **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# tls trust-point local tp1 remote tp2`	Enters the TLS configuration mode. Configures the local and remote trustpoints.
Step 5	**bridge-group** *SDN-id* **bridge-domain** *switch-id* **Example:** `RP/0/RSP0/CPU0:router (config-openflow) # bridge-group SDN-1 bridge-domain of2`	Configures the bridge-domain for the openflow switch. For layer2, the bridge-domain can be configured in the openflow switch and the interfaces of the bridge-domain will be learnt by the openflow switch.
Step 6	**controller ipv4** *ip-address* **security [tls | none]** **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# controller ipv4 5.0.1.1 port 6633 security tls`	Configures the Openflow controller for the logical switch. Configures the Openflow controller for the logical switch. Once the **controller** command is entered, a connection to the OpenFlow controller is started for the logical switch. The **tls** keyword enables the TLS connection, whereas the **none** keyword enables the TCP connection. **Note** The OpenFlow Agent can connect to a single Controller or up to 8 Controllers. Repeat this step if you need to configure additional Controllers. An openflow switch can communicate to multiple controllers (the support for high-availability is a controller functionality).
Step 7	**commit** **Example:** `RP/0/RSP0/CPU0:router(logical-switch)# commit`	Adds the Layer 2 logical switch configuration for the OpenFlow agent to the running configuration.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions:

	Command or Action	Purpose
		• **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Repeat these steps to configure another logical switch for the OpenFlow Agent.

Configuring a Layer 2_Layer 3 Logical Switch for the OpenFlow Agent

SUMMARY STEPS

1. **configure**
2. **openflow**
3. **switch** *switch -id* **pipeline** *pipeline-number*
4. **tls trust-point local** *local-tp-name* **remote** *remote-tp-name*
5. **bridge-group** *SDN-id* **bridge-domain** *switch-id*
6. **controller ipv4** *ip-address* **security [tls | none]**
7. **commit**
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**openflow** **Example:** `RP/0/RSP0/CPU0:router(config)# openflow`	Enters the openflow configuration mode.
Step 3	**switch** *switch -id* **pipeline** *pipeline-number* **Example:** `RP/0/RSP0/CPU0:router(config-openflow)# switch 1 pipeline 130`	Enters the logical switch configuration mode. For L2_L3 switch, the pipeline number is 130.

	Command or Action	Purpose
Step 4	**tls trust-point local** *local-tp-name* **remote** *remote-tp-name* **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# tls trust-point local tp1 remote tp2`	Enters the TLS configuration mode. Configures the local and remote trustpoints.
Step 5	**bridge-group** *SDN-id* **bridge-domain** *switch-id* **Example:** `RP/0/RSP0/CPU0:router (config-openflow) # bridge-group SDN-1 bridge-domain of2`	Configures a bridge-domain for the openflow switch.
Step 6	**controller ipv4** *ip-address* **security [tls | none]** **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# controller ipv4 5.0.1.1 port 6633 security tls`	Configures the Openflow controller for the logical switch. Configures the Openflow controller for the logical switch. Once the **controller** command is entered, a connection to the OpenFlow controller is started for the logical switch. The **tls** keyword enables the TLS connection, whereas the **none** keyword enables the TCP connection. **Note** The OpenFlow Agent can connect to a single Controller or up to 8 Controllers. Repeat this step if you need to configure additional Controllers. An openflow switch can communicate to multiple controllers (the support for high-availability is a controller functionality).
Step 7	**commit** **Example:** `RP/0/RSP0/CPU0:router(logical-switch)# commit`	Adds the Layer 2 logical switch configuration for the OpenFlow agent to the running configuration.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Repeat these steps to configure another logical switch for the OpenFlow Agent.

Configuring a Layer 3_VRF Logical Switch for the OpenFlow Agent

SUMMARY STEPS

1. **configure**
2. **openflow**
3. **switch** *switch -id* **pipeline** *pipeline-number*
4. **vrf IPv4**
5. **tls trust-point local** *local-tp-name* **remote** *remote-tp-name*
6. **controller ipv4** *ip-address* **security [tls | none]**
7. **commit**
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**openflow** **Example:** `RP/0/RSP0/CPU0:router(config)# openflow`	Enters the openflow configuration mode.
Step 3	**switch** *switch -id* **pipeline** *pipeline-number* **Example:** `RP/0/RSP0/CPU0:router(config-openflow)# switch 1 pipeline 131`	Enters the logical switch configuration mode. For L3_V4(VRF) switch, the pipeline number is 131.
Step 4	**vrf IPv4** **Example:** `RP/0/RSP0/CPU0:router(config)# vrf IPv4`	VRF configuration. All the interfaces belonging to IPv4 VRF will be learnt by the openflow switch.
Step 5	**tls trust-point local** *local-tp-name* **remote** *remote-tp-name* **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# tls trust-point local tp1 remote tp2`	Enters the TLS configuration mode. Configures the local and remote trustpoints.
Step 6	**controller ipv4** *ip-address* **security [tls | none]** **Example:**	Configures the Openflow controller for the logical switch. Configures the Openflow controller for the logical switch. Once the **controller** command is entered, a connection to the OpenFlow controller is started for the logical switch.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-openflow-switch)#` **`controller ipv4 5.0.1.1 port 6633 security tls`**	**Note** The OpenFlow Agent can connect to a single Controller or up to 8 Controllers. Repeat this step if you need to configure additional Controllers.
Step 7	**commit** **Example:** `RP/0/RSP0/CPU0:router(logical-switch)# commit`	Adds the Layer 2 logical switch configuration for the OpenFlow agent to the running configuration.
Step 8	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Repeat these steps to configure another logical switch for the OpenFlow Agent.

Configuring a Layer 3_Dual-stack Logical Switch for the OpenFlow Agent

SUMMARY STEPS

1. **configure**
2. **openflow**
3. **switch** *switch -id* **pipeline** *pipeline-number*
4. **interface** *type interface-path-id*
5. **tls trust-point local** *local-tp-name* **remote** *remote-tp-name*
6. **bridge-group** *SDN-id* **bridge-domain** *switch-id*
7. **controller ipv4** *ip-address* **security [tls | none]**
8. **commit**
9. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**openflow** **Example:** `RP/0/RSP0/CPU0:router(config)# openflow`	Enters the openflow configuration mode.
Step 3	**switch** *switch -id* **pipeline** *pipeline-number* **Example:** `RP/0/RSP0/CPU0:router(config-openflow)# switch 1 pipeline 132`	Enters the logical switch configuration mode. For L3_DS switch, the pipeline number is 132.
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-openflow)# interface Bundle-Ether2.1`	Interface configuration. **Note** VRFs can be configured here. Both IPv4 abd IPv6 VRFs are supported.
Step 5	**tls trust-point local** *local-tp-name* **remote** *remote-tp-name* **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# tls trust-point local tp1 remote tp2`	Enters the TLS configuration mode. Configures the local and remote trustpoints.
Step 6	**bridge-group** *SDN-id* **bridge-domain** *switch-id* **Example:** `RP/0/RSP0/CPU0:router (config-openflow) # bridge-group SDN-1 bridge-domain of2`	
Step 7	**controller ipv4** *ip-address* **security [tls | none]** **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# controller ipv4 5.0.1.1 port 6633 security tls`	Configures the Openflow controller for the logical switch. Configures the Openflow controller for the logical switch. Once the **controller** command is entered, a connection to the OpenFlow controller is started for the logical switch. **Note** The OpenFlow Agent can connect to a single Controller or up to 8 Controllers. Repeat this step if you need to configure additional Controllers.
Step 8	**commit** **Example:** `RP/0/RSP0/CPU0:router(logical-switch)# commit`	Adds the Layer 2 logical switch configuration for the OpenFlow agent to the running configuration.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Repeat these steps to configure another logical switch for the OpenFlow Agent.

Enabling TLS

SUMMARY STEPS

1. **configure**
2. **openflow switch** *logical-switch-id*
3. **tls trust-point local** *local-tp-name* **remote** *remote-tp-name*
4. **commit**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**openflow switch** *logical-switch-id* **Example:** `RP/0/RSP0/CPU0:router(config)# openflow switch 100`	Enters the OpenFlow logical switch configuration mode.
Step 3	**tls trust-point local** *local-tp-name* **remote** *remote-tp-name* **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# tls trust-point local tp1 remote tp2`	Enters the TLS configuration mode. Configures the local and remote trustpoints.
Step 4	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# commit`	Adds the logical switch configuration for the OpenFlow agent to the running configuration.

	Command or Action	Purpose
Step 5	**end** **Example:** `RP/0/RSP0/CPU0:router(config-openflow-switch)# end`	Exits logical switch configuration mode and enters EXEC mode.

Configuring NetFlow for the OpenFlow Agent

SUMMARY STEPS

1. **configure**
2. **flow exporter-map** *fem-name*
3. **destination** *location*
4. **version v9**
5. **commit**
6. **exit**
7. **flow monitor-map** *map-name*
8. **record ipv4**
9. **exporter** *map-name*
10. **cache entries** *number*
11. **cache timeout** {**active** *timeout-value* | **inactive** *timeout-value* | **update** *timeout-value*}
12. **commit**
13. **exit**
14. **sampler-map** *map-name*
15. **random 1 out-of** *sampling-interval*
16. **commit**
17. **exit**
18. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**flow exporter-map** *fem-name* **Example:** `RP/0/RSP0/CPU0:router(config)# flow exporter-map fem`	Enters flow exporter map configuration mode. **Note** A single flow monitor map can support up to eight exporters.

	Command or Action	Purpose
Step 3	**destination** *location* **Example:** `RP/0/RSP0/CPU0:router(config-fem)# destination 10.0.1.2`	Configures the export destination for the flow exporter map. The destination **location** argument can be a hostname or an IP address.
Step 4	**version v9** **Example:** `RP/0/RSP0/CPU0:router(config-fem)# version v9`	Specifies export version parameters and enters the flow exporter map version configuration mode.
Step 5	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-fem-ver)# commit`	Commits the configuration changes to running to the running configuration.
Step 6	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-fem-ver)# exit`	Exits flow exporter map version configuration mode and enters global configuration mode.
Step 7	**flow monitor-map** *map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# flow monitor-map mmap`	Creates a monitor map and configures a monitor map name and enters flow monitor map configuration mode
Step 8	**record ipv4** **Example:** `RP/0/RSP0/CPU0:router(config-fmm)# record ipv4`	Configures the flow record map name for IPv4. By default, the originating autonomous system (AS) numbers are collected and exported.
Step 9	**exporter** *map-name* **Example:** `RP/0/RSP0/CPU0:router(config-fmm)# exporter fmap`	Associates an exporter map with a monitor map. **Note** A single flow monitor map can support up to eight exporters.
Step 10	**cache entries** *number* **Example:** `RP/0/RSP0/CPU0:router(config-fmm)# cache entries 4096`	(Optional) Configures the number of entries in the flow cache. Replace the number argument with the number of flow entries allowed in the flow cache, in the range from 4096 through 1000000. The default number of cache entries is 65535.
Step 11	**cache timeout** {**active** *timeout-value* | **inactive** *timeout-value* | **update** *timeout-value*} **Example:** `RP/0/RSP0/CPU0:router(config-fmm)# cache timeout active 10`	(Optional) Configures the active, inactive, or update flow cache timeout value. • The default timeout value for the inactive flow cache is 15 seconds. • The default timeout value for the active flow cache is 1800 seconds. • The default timeout value for the update flow cache is 1800 seconds.

	Command or Action	Purpose
		Note The **update** keyword and *timeout-value* argument are used for permanent caches only. It specifies the timeout value that is used to export entries from permanent caches. In this case, the entries are exported but remain the cache.
Step 12	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-fmm)# commit`	Commits the configuration changes to running to the running configuration.
Step 13	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-fmm)# exit`	Exits flow monitor map version configuration mode and enters global configuration mode.
Step 14	**sampler-map** *map-name* **Example:** `RP/0/RSP0/CPU0:router(config)# sampler-map`	Creates a sampler map and enters sampler map configuration mode. **Note** When configuring a sampler map, be aware that NetFlow supports policing at a rate of 35,000 packets per second per direction for each individual line card.
Step 15	**random 1 out-of** *sampling-interval* **Example:** `RP/0/RSP0/CPU0:router(config-sm)# random 1 out-of 65535`	Configures the sampling interval to use random mode for sampling packets. For the *sampling-interval* argument, specify a number from 1 to 65535.
Step 16	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-sm)# commit`	Commits the configuration changes to running to the running configuration.
Step 17	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-sm)# exit`	Exits sampler map version configuration mode and enters global configuration mode.
Step 18	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

Go to the "Associating the OpenFlow Agent Logical Switch with NetFlow" section to complete the second part of this configuration.

Configuration Examples: Openflow

Attaching a bridge domain to an Openflow Switch: Examples

- Attaching a L2-only Openflow switch

```
openflow
switch 1 pipeline 129
  tls trust-point local tp1 remote tp1
  bridge-group SDN-2 bridge-domain OF-2
  controller ipv4 5.0.1.200 port 6653 security tls
```

- Attaching a L2_L3 Openflow switch

```
openflow
switch 1 pipeline 130
  tls trust-point local tp1 remote tp1
  bridge-group SDN-2 bridge-domain OF-2
  controller ipv4 5.0.1.200 port 6653 security tls
```

- L3_V4 switch can be attached either to a VRF or directly to layer 3 interfaces under global VRF. In case of VRF, all the interfaces in that VRF become part of the OpenFlow switch.

```
openflow
switch l1 pipeline 131
  vrf IPv4
  controller ipv4 5.0.1.200 port 6653 security none
 !
```

- L3_DS switch can be attached either to a VRF or directly to layer 3 interfaces under global VRF.

```
openflow
switch l2 pipeline 132
  vrf IPv4
  controller ipv4 5.0.1.200 port 6653 security none
 !
```

OpenFlow Agent with NetFlow Collection and Analytics Configuration: Example

The following example describes the NetFlow exporter map configuration for the OpenFlow logical switch.

```
Device> enable
Device# configure terminal
Device(config)# flow exporter-map fem
Device(config-fem)# destination 10.0.1.2
Device(config-fem)# version v9
```

```
Device(config-fem-ver)# commit
Device(config-fem-ver)# exit
```

The following example describes the NetFlow monitor map configuration for the OpenFlow logical switch.

```
Device(config)# flow monitor-map mmap
Device(config-fmm)# record ipv4
Device(config-fmm)# exporter fmap
Device(config-fmm)# cache entries 4096
Device(config-fmm)# commit
Device(config-fmm)# exit
```

The following example describes the NetFlow sampler map configuration for the OpenFlow logical switch.

```
Device(config)# sampler-map
Device(config-sm)# random 1 out-of 65535
Device(config-sm)# commit
Device(config-sm)# exit
```

The following example describes how the OpenFlow Agent logical switch is configured so that the NetFlow collection and analytics are associated with it.

```
Device(config)# openflow switch 100 netflow
Device(logical-switch)# flow monitor mmap sampler smap
Device(logical-switch)# interface GigabitEthernet0/1/0/6
Router(logical-switch)# controller 10.0.1.2 port 6633
Device(logical-switch)# commit
Device(logical-switch)# end
```

The following example describes **show** command output for an OpenFlow Agent logical switch that is configured with NetFlow collection and analytics.

```
Device# show openflow switch 100
Fri Jan 25 14:29:21.078 UTC

Logical Switch Context
         Id:                    100
         Switch type:           Netflow
         Layer:                 NONE
         Signal version:        Openflow 1.0
         Data plane:            secure
         Fallback:              normal
         Config state:          no-shutdown
         Working state:         enabled
         TLS version:           NONE
         TLS private key:       none:none
         TLS private key file:  NONE
         TLS certificate file:  NONE
         Controller:            10.0.1.2:6633, last alive ping: 2013-01-25 14:29:20
         Netflow Monitor:       mmap
         Netflow Sampler:       smap
         Loopback i/f:          <none>
         Loopback addr:         <none>
         Interfaces:
                GigabitEthernet0/1/0/6

Device# show openflow switch 100 flows
```

```
Fri Jan 25 14:29:24.787 UTC

Logical Openflow Switch [100]:
NXST_FLOW reply (xid=0x0):
cookie=0x0, duration=204.729s, table=0, n_packets=0, n_bytes=0, priority=500 actions=netflow


Switch flow count: 1

Device# show openflow switch 100 controllers
Fri Jan 25 14:29:28.660 UTC

Logical Openflow Switch [100]:
        Controller [tcp:10.0.1.2:6633]
           role                :  Other
           connected           :  Yes
           state               :  ACTIVE
           sec_since_connect   :  487
```

Usecase for Layer2

The Scenario: Enterprise Data Center needs to perform data backup to multiple other backup sites based on the Traffic flow. The Main DC is in Vlan 100 and Backup sites are at VLAN 1000,1001,1002. These Sites are interconnected through L2VPN.

The Solution: Openflow, we can match any Layer 2 header field (in this example we have taken priority bits) and steer the traffic to go on any L2 interconnect and also rewrite the VLANs appropriately.

Usecase for Layer3

The Scenario: Three different flows from 3 different sites connected to PE1 are trying to send 350 mbps of traffic each to PE2. The bandwidth of the shortest link, Path-2 (between PE1 and PE2) is only 1 Gigabit. Hence Path-2 gets congested as soon as the third site begins to send traffic.

The Solution: Openflow controller can be used to install rules on PE1:

- Match on Flow 1 (destined to Video server) and redirect traffic to Path-2
- Match on Flow 2 (destined to Web server) and redirect traffic to Path-1
- Match on Flow 3 (destined to File transfer server) and redirect traffic to Path-3

The Inference: Effectively utilizing the network bandwidth by redirecting destination specific traffic using OpenFlow rules.