Virtual LANs in Layer 2 VPNs

This chapter provides conceptual information for Virtual LANs in general and configuration information of layer 2 VLANs on Cisco 8000 Series Router.

- Introduction to Virtual LANs in Layer 2 VPNs, on page 1
- L2 VLAN Subinterface Encapsulation and Rewrite, on page 5
- VLAN Subinterface, on page 10
- Ethernet Flow Point, on page 13

Introduction to Virtual LANs in Layer 2 VPNs

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The support for Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation is now extended to all systems in the Cisco 8000 Series Routers.

Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation	Release 24.1.1	We have optimized VLAN implementation by enabling service providers to: • expand VLAN space to segregate their networks for customers with multiple VLANs and overlapping VLAN IDs. • enhance service mapping for efficiently differentiating data packets and applying QoS policies based on users and services. Such optimization is possible because this release supports Dot1Q Q-in-Q (0x8100/0x8100) encapsulation for VLAN subinterfaces. This involves configuring these subinterfaces to add an outer 802.1Q tag to packets that are already carrying an 802.1Q VLAN tag. The feature introduces these changes: **CLI:** New L2VPN commands • **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* • **rewrite ingress tag** **YANG Data Model:** • New XPaths for `openconfig-interfaces.yang` (see GitHub, YANG Data Models Navigator) This feature is supported on Cisco 8000 series routers that are based on the Q200 silicon chip application-specific integrated circuit (ASIC).

The Layer 2 Virtual Private Network (L2VPN) feature enables Service Providers (SPs) to provide L2 services to geographically disparate customer sites.

A virtual local area network (VLAN) is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different LAN segments. The IEEE's 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames.

VLANs are very useful for user and host management, bandwidth allocation, and resource optimization. Using VLANs addresses the problem of breaking large networks into smaller parts so that broadcast and multicast traffic does not consume more bandwidth than necessary. VLANs also provide a higher level of security between segments of internal networks.

The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames. Cisco IOS XR software supports VLAN subinterface configuration on four hundred Gigabit Ethernet and one hundred Gigabit Ethernet interfaces.

The configuration model for configuring VLAN Attachment Circuits (ACs) is similar to the model used for configuring basic VLANs, where the user first creates a VLAN subinterface, and then configures that VLAN in subinterface configuration mode. To create an Attachment Circuit, you need to include the **l2transport** keyword in the interface command string to specify that the interface is a L2 interface.

- Basic Dot1Q Attachment Circuit—The Attachment Circuit covers all frames that are received and sent with a specific VLAN tag.

- Q-in-Q Attachment Circuit—The AC covers all frames received and sent with a specific outer VLAN tag and a specific inner VLAN tag. Q-in-Q is an extension to basic dot1q and uses a stack of two tags.

Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation

802.1Q tunneling (or Q-in-Q), as defined by IEEE 802.1ad, extends VLAN capacity by appending an extra 802.1Q tag to packets that are already 802.1Q-tagged. Q-in-Q encapsulation, also known as stacked VLAN tagging or double VLAN is a technique used in networking to add an extra layer of VLAN tagging to Ethernet frames.

In a standard VLAN configuration, each Ethernet frame has a single VLAN tag that identifies the VLAN to which it belongs. Q-in-Q adds another layer of VLAN tagging, allowing for the creation of multiple VLAN domains within a larger network. Q-in-Q enables a more scalable VLAN implementation. The outer VLAN tag represents the service provider VLAN, and the inner VLAN tag represents the customer VLAN. This enables the service provider to support multitenancy and manage a large number of customers with overlapping VLAN IDs over the same carrier network.

In addition to the advantages associated with expanding VLAN space, Q-in-Q tunneling also facilitates service mapping. The use of inner and outer VLAN tags allows the differentiation of packets based on users and services.

In Q-in-Q encapsulation, there are two levels of VLAN tags:

- **Outer VLAN Tag**:
  - Identifies the VLAN of the service provider network.
  - Added by the service provider when the frame enters its network.
  - The EtherType can be either dot1q (0x8100), dot1q (0x9100), or dot1ad (0x88A8), depending on the configuration or platform.
- **Inner VLAN Tag**:
  - Identifies the VLAN of the customer within the service provider's VLAN.
  - Added by the customer network.
  - The EtherType must be dot1q (0x8100).

Before a sub-interface is allowed to pass traffic, it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet subinterfaces always default to the 802.1Q VLAN encapsulation. However, the VLAN identifier must be explicitly defined.

Configure Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation

To configure a dot1q Q-in-Q tunneling for VLAN sub-interfaces with an outer tag of 0x8100, use the following example configuration:

```
Router#configure
Router(config)#interface TenGigE 0/0/0/1.102 l2transport
Router(config-subif)#encapsulation dot1q 200 second-dot1q 201
Router(config-subif)#commit
Router(config-subif)#exit
Router(config)#exit
```

Running Configuration

```
configure
 interface TenGigE 0/0/0/1.102
  l2transport
  encapsulation dot1q 200 second-dot1q 201
 !
!
```

Verification

Verify that the VLAN subinterface is in Q-in-Q mode:

```
Router# show interfaces TenGigE 0/0/0/1.102
Wed Sep 8 14:50:15.691 UTC
HundredGigE0/0/0/1.102 is up, line protocol is up
Interface state transitions: 1
Hardware is VLAN sub-interface(s), address is 0029.c201.3f0c
Internet address is 40.40.50.1/24
MTU 1522 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation 802.1Q-802.1Q Virtual LAN,
Last link flapped 00:01:25
ARP type ARPA, ARP timeout 04:00:00
Last input never, output never
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
0 packets output, 0 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

Configure Dot1Q Q-in-Q (0x8100/0x8100) Tag Rewrite for VLAN Sub-interface

Optional: To add or modify double Dot1q Q-in-Q VLAN tags on Layer 2 Ethernet frames with an outer tag of 0x8100, use the following example configurations.

```
Router#configure
Router(config)#interface TenGigE 0/0/0/1.102 l2transport
Router(config-subif)#encapsulation dot1q 200 second-dot1q 201
Router(config-subif)#rewrite ingress tag pop 2 symmetric
Router(config-subif)#commit
Router(config-subif)#exit
Router(config)#exit
```

Running Configuration

```
/* Configure Dot1Q Q-in-Q Tag Rewrite: Pop 2 */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 200 dot1q 201
  rewrite ingress tag pop 2 symmetric
 !
!

/* Configure Dot1Q Q-in-Q Tag Rewrite: Push */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 200 dot1q 201
  rewrite ingress tag push dot1q 200 second-dot1q 201 symmetric
```

```
 !
!

/* Configure Dot1Q Q-in-Q Tag Rewrite: Translate 1-to-2 */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 200 dot1q 201
  rewrite ingress tag translate 1-to-2 dot1q 200 second-dot1q 201 symmetric
 !
!

/* Configure Dot1Q Q-in-Q Tag Rewrite: Translate 2-to-2 */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 200 dot1q 201
  rewrite ingress tag translate 2-to-2 dot1q 200 second-dot1q 201 symmetric
 !
!
```

Double-Tagged 802.1ad Encapsulation Options for Layer 2 and Layer 3 Physical and Bundle Subinterfaces

Subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow for segregation of traffic into separate logical channels on a single hardware interface as well as allowing for better utilization of the available bandwidth on the physical interface.

Before a subinterface is allowed to pass traffic it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet subinterfaces always default to the 802.1Q VLAN encapsulation. However, the VLAN identifier must be explicitly defined.

L2 VLAN Subinterface Encapsulation and Rewrite

Table 2: Feature History Table

Feature Name	Release Information	Feature Description

L2 VLAN Subinterface Encapsulation and Rewrite	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now use the VLAN Subinterface Encapsulation and Rewrite operations to: • Configure exact matching for all single-tagged encapsulations. • Support legacy Q-in-Q encapsulation 0x9100/0x8100. • Enable priority tagged traffic to map to the specified interface. The feature introduces these changes: **CLI:** • **dot1q tunneling ethertype 0x9100** • **hw-module profile encap-exact** • **encapsulation dot1ad priority-tagged** • **encapsulation dot1q priority-tagged** • **rewrite ingress tag** **YANG Data Model:** • `Cisco-IOS-XR-um-8000-hw-module-profile-cfg` (see GitHub, YANG Data Models Navigator)

Configure exact matching for single-tagged encapsulations

The hardware module profile encap-exact command is used to configure exact matching for all single-tagged encapsulations (for the specified interface, interface type, or location). All sub-interfaces associated with the configured interface uses exact matching.

Exact matching applies exclusively to single-tagged encapsulations.

The purpose of exact matching is to ensure that double-tagged traffic does not match single-tagged encapsulations (sub-interfaces). With exact matching, incoming traffic must have exactly the same number of recognized tags (1) to match single-tagged encapsulations.

Note For the hardware module profile encap-exact configuration to take effect, you must reload the line card or router.

The **hardware module profile encap-exact** encapsulation can be configured in the following locations:

- All interfaces in the system

- All interfaces on a device or line card
- All bundle interfaces in the system
- A specific interface in the system

Configuration Examples

To configure exact matching for all main interfaces in the system:

```
Router#configure
Router(config)#hw-module profile encap-exact location all
```

To configure exact matching for the specified location:

```
Router#configure
Router(config)#hw-module profile encap-exact location <node-id>
```

To configure exact matching for all main virtual (Bundle) interfaces in the system:

```
Router#configure
Router(config)#hw-module profile encap-exact location all-virtual
```

To configure exact matching for a specific interface using the interface name (Not a Bundle):

```
Router#configure
Router(config)#hw-module profile encap-exact interface <intf>
```

Running Configuration

```
configure
hw-module profile encap-exact location all
Wed Nov 20 16:30:52.007 EST
In order to activate/deactivate this profile, you must manually reload the chassis/all line
 cards
```

Verification

```
/* Before Reload */

Router# show hw-module profile encap-exact
Wed Nov 20 16:31:09.499 EST
-------------------------------------------------------------------
Knob                          Status          Applied   Action
-------------------------------------------------------------------
Encap Exact                 Configured      No        Reload
```

Before reload, the configuration is not applied. You must reload for the configuration to take effect.

```
/* After Reload */

Router# show hw-module profile encap-exact
Wed Nov 20 16:48:57.285 EST
-------------------------------------------------------------------
Knob                          Status          Applied   Action
-------------------------------------------------------------------
Encap Exact                 Configured      Yes       None
```

Support legacy QinQ encapsulation 0x9100/0x8100

To support legacy QinQ encapsulation 0x9100/0x8100, use the **dot1q tunneling ethertype 0x9100** command. This configuration command causes the 0x8100 QinQ encapsulation type (**encapsulation dot1q <x>**

second-dot1q <y>) to use 0x9100 as the outer ethertype, instead of 0x8100, for all sub-interfaces under the main interface where it is configured.

Configuration Example

The below configuration example shows how the legacy QinQ encapsulation 0x9100/0x8100 is used as the outer ethertype for the interface FH0/0/0/3.1.

```
Router#configure
Router(config)#interface FH0/0/0/3
Router(config-subif)#dot1q tunneling ethertype 0x9100
Router(config-subif)#interface FH0/0/0/3.1 l2transport
Router(config-subif)#encapsulation dot1q 500 second-dot1q 600
Router(config-subif)#commit
Router(config-subif)#exit
Router(config)#exit
```

Running Configuration

```
configure
interface FH0/0/0/3
    dot1q tunneling ethertype 0x9100
interface FH0/0/0/3.1 l2transport
    encapsulation dot1q 500 second-dot1q 600
```

Verification

```
Router# show run int FH0/0/0/3.1
Wed Nov 20 16:21:21.747 EST
interface FourHundredGigE0/0/0/3.1 l2transport
encapsulation dot1q 500 second-dot1q 600
!

Router# show run int FH0/0/0/3
Wed Nov 20 16:21:34.932 EST
interface FourHundredGigE0/0/0/3
dot1q tunneling ethertype 0x9100
!

Router# show int FH0/0/0/3.1
Wed Nov 20 16:21:44.926 EST
FourHundredGigE0/0/0/3.1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 0075.f409.6818
  Layer 2 Transport Mode
  MTU 1522 bytes, BW 400000000 Kbit (Max: 400000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 500
    Inner Match: Dot1Q VLAN 600
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
  Last link flapped 00:00:34
  Last input never, output never
  Last clearing of "show interface" counters never
     0 packets input, 0 bytes
     0 input drops, 0 queue drops, 0 input errors
     0 packets output, 0 bytes
     0 output drops, 0 queue drops, 0 output errors

Router# show int FH0/0/0/3
```

```
Wed Nov 20 16:22:04.539 EST
FourHundredGigE0/0/0/3 is up, line protocol is up
  Interface state transitions: 1
  Hardware is FourHundredGigE, address is 0075.f409.6818 (bia 0075.f409.6818)
  Internet address is Unknown
  MTU 1514 bytes, BW 400000000 Kbit (Max: 400000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation ARPA,
  Full-duplex, 400000Mb/s, link type is force-up
  output flow control is off, input flow control is off
  Carrier delay (up) is 200 msec
  loopback not set,
  Last link flapped 00:00:53
  Last input never, output never
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     1 carrier transitions
```

Configure Priority-tagged traffic

This configuration allows priority-tagged traffic to map to the specified interface. The priority-tagged traffic has a VLAN tag with VLAN ID of 0.

Configuration Example

```
Router#configure
Router(config)#interface TenGigE0/0/0/0.1 l2transport
Router(config-subif)#encapsulation dot1q priority-tagged
Router(config-subif)#commit
Router(config-subif)#exit
Router(config)#exit
```

Verification

You can verify the priority tagged configuration by using the **show interfaces** command.

```
Router# show int TenGigE0/0/0/0.1
Fri Dec 13 15:22:08.649 EST
TenGigE0/0/0/0.1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 20db.ea1a.1a01
  Layer 2 Transport Mode
  MTU 1518 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q priority tagged,
    Outer Match: Dot1Q VLAN Priority-tagged
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
  Last input never, output never
  Last clearing of "show interface" counters never
     0 packets input, 0 bytes
     0 input drops, 0 queue drops, 0 input errors
```

```
0 packets output, 0 bytes
0 output drops, 0 queue drops, 0 output errors
```

Supported Encapsulation

Table 3: 802.1ad Encapsulation Support for Layer 2 Interfaces and subinterfaces

Interface Type	Encapsulation	Standard	Support Status
Layer 2 interface Layer 2 subinterface Layer 2 bundle subinterface	Single-Tag Encapsulation	dot1ad	Supported (From Cisco IOS XR Software Release 24.4.1 onwards)
		dot1q	Supported.
		default	Supported.
		untagged	Supported.
		dot1q priority-tagged	Supported (From Cisco IOS XR Software Release 24.4.1 onwards)
		dot1ad priority-tagged	Supported (From Cisco IOS XR Software Release 24.4.1 onwards)
	Double-Tag Encapsulation	dot1ad <> dot1q<>	Supported.
		dot1q <> dot1q<>	[1]Supported.

[1] The **encapsulation dot1q <x> second-dot1q <y>** encapsulation type is supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.

VLAN Subinterface

Sub-interfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow segregation of traffic into separate logical channels on a single hardware interface. Also, helps in better utilization of the available bandwidth on the physical interface.

Sub-interfaces are distinguished from one another by adding an extension on the end of the interface name and designation. For instance, the Ethernet sub-interface 23 on the physical interface designated HundredGigE 0/1/0/0 would be indicated by HundredGigE 0/1/0/0.23.

Before a sub-interface is allowed to pass traffic, it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet sub-interfaces always default to the 802.1Q VLAN encapsulation. However, the VLAN identifier must be explicitly defined.

The sub-interface Maximum Transmission Unit (MTU) is inherited from the physical interface with 4 bytes allowed for the 802.1Q VLAN tag.

The Basic dot1q Attachment Circuit mode of VLAN sub-interface configuration is supported:

To configure a basic dot1q Attachment Circuit, use this encapsulation mode:

encapsulation dot1q *vlan_id*

You can configure **encapsulation default** on L2 subinterfaces.

For example,

```
configure
 interface HundredGigE 0/0/0/10.1
  l2transport
   encapsulation default
```

VLAN List and VLAN Range

VLANs separated by comma are called a VLAN list. VLANs separated by dashes are called a VLAN range. For more information about VLAN list and range, see L2 Interface VLAN Encapsulation using VLAN Range and List.

Starting from Cisco IOS XR Software Release 24.4.1 onwards, VLAN list and VLAN range are supported for the following encapsulation types:

- **encapsulation dot1ad**
- **encapsulation dot1ad dot1q**
- **encapsulation dot1q**
- **encapsulation dot1q second-dot1q**

Configure VLAN SubInterface

Configuring VLAN subinterface involves:

- Creating a subinterface
- Enabling L2 transport mode on the newly created subinterface
- Defining the matching criteria (encapsulation mode) to be used in order to map ingress frames on an interface to the appropriate service instance

Configuration Example

Configuration of basic dot1q attachment circuit:

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/10.1 l2transport
Router(config-if)# encapsulation dot1q 10
Router(config-if)# no shutdown
```

Configuration of double-tagged dot1ad and dot1q attachment circuit:

```
Router# configure
Router(config)# interface HundredGigE 0/0/0/10.1 l2transport
Router(config-if)# encapsulation dot1ad 200 dot1q 201
Router(config-if)# no shutdown
```

Running Configuration

```
configure
 interface HundredGigE 0/0/0/10.1
```

```
  l2transport
  encapsulation dot1q 10
 !
!


configure
 interface HundredGigE 0/0/0/10.1 l2transport
  encapsulation dot1ad 200 dot1q 201
!
 !
```

Verification

Verify that the VLAN subinterface is active in a single-tag scenario.

```
Router# show interfaces hundredGigE 0/0/0/29.300
Wed Sep 8 14:55:25.812 UTC
HundredGigE0/0/0/29.300 is up, line protocol is up
Interface state transitions: 1
Hardware is VLAN sub-interface(s), address is 0029.c201.3f0c
Internet address is Unknown
MTU 1518 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation 802.1Q Virtual LAN, VLAN Id 300, loopback not set,
Last link flapped 00:00:19
Last input never, output never
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
0 packets output, 0 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

Verify that the VLAN subinterface is active in a double-tag scenario.

```
Router# show interface HundredGigE 0/0/0/29.200
Wed Sep 8 14:50:15.691 UTC
HundredGigE0/0/0/29.200 is up, line protocol is up
Interface state transitions: 1
Hardware is VLAN sub-interface(s), address is 0029.c201.3f0c
Internet address is 40.40.50.1/24
MTU 1522 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
reliability 255/255, txload 0/255, rxload 0/255
Encapsulation 802.1ad-802.1Q Virtual LAN,
Last link flapped 00:01:25
ARP type ARPA, ARP timeout 04:00:00
Last input never, output never
Last clearing of "show interface" counters never
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
0 packets input, 0 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
Received 0 broadcast packets, 0 multicast packets
0 packets output, 0 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

Ethernet Flow Point

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Ethernet Flow Point	Release 24.4.1	Introduced in this release on: Fixed Systems (8700) (select variants only*) * The Ethernet Flow Point functionality is now extended to the Cisco 8712-MOD-M routers.
Ethernet Flow Point	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) * The Ethernet Flow Point functionality is now extended to these fixed systems and line cards: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Ethernet Flow Point	Release 24.2.11	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) An Ethernet Flow Point (EFP) enhances traffic management and network efficiency by classifying and processing Layer 2 traffic under a physical or bundle interface based on specific filters, such as VLAN tags. This logical sub-interface allows for precise traffic classification and manipulation, including changing VLAN IDs, adding or removing VLAN tags, and modifying ethertypes upon ingress. * This functionality is now extended to routers with the 88-LC1-36EH line cards.

An Ethernet Flow Point (EFP) is a Layer 2 logical sub-interface used to classify traffic under a physical or a bundle interface. An EFP is defined by a set of filters (a set of entries) that are applied to all the ingress traffic to classify the frames that belong to a particular EFP. Each entry usually contains 0, 1 or 2 VLAN tags. You can specify a VLAN or QinQ tagging to match against on ingress. A packet that starts with the same tags as an entry in the filter is said to match the filter; if the start of the packet does not correspond to any entry in the filter, then the packet does not match the filter.

All traffic on ingress are processed by that EFP if a match occurs, and this can in turn change VLAN IDs, add or remove VLAN tags, and change ethertypes. After the frames are matched to a particular EFP, any appropriate feature (such as, any frame manipulations specified by the configuration as well as things such as QoS) can be applied.

Note The following terms are used interchangeably throughout this document:

- VLAN AC
- L2 interface
- EFP

Benefits of EFP

- Identify all frames that belong to a particular flow on a given interface.
- Perform VLAN header rewrites.
- Add features to the identified frames.
- Optionally define how to forward the identified frames in the data path.

The following are the components of EFP:

Identify Frames of an EFP

The EFP identifies frames belonging to a particular flow on a given port, independent of their Ethernet encapsulation. An EFP can flexibly map frames into a flow or EFP based on the fields in the frame header. The frames can be matched to an EFP using VLAN tag(s).

The frames cannot be matched to an EFP through this:

- Any information outside the outermost Ethernet frame header and its associated tags such as
  - IPv4, IPv6, or MPLS tag header data
  - C-DMAC, C-SMAC, or C-VLAN

VLAN Tag Identification

Below table describes the different encapsulation types and the EFP identifier corresponding to each.

Encapsulation Type	EFP Identifier
Single outer most tag	The tag must use EtherType 0x8100 or 0x88a8.
Two outer most tags	The outer-most tag can use EtherType 0x8100, 0x9100, or 0x88A8, depending on the platform or configuration. The second outer-most VLAN tag must use EtherType 0x8100.

Apply Features

After the frames are matched to a particular EFP, any appropriate features can be applied. In this context, “features” means any frame manipulations specified by the configuration. For example, QoS. You can apply QoS policy on L2 interfaces. For more information about QoS polices that are supported on L2 interfaces, see *Modular QoS Configuration Guide for Cisco 8000 Series Routers*.

The L2 header encapsulation modification is the L2 interface VLAN tag rewrite that is applied on an EFP, which is part of the Ethernet infrastructure.

Encapsulation Modifications

EFP supports these L2 header encapsulation modifications on both ingress and egress:

- Push 1 VLAN tag - push an Ethertype 0x8100 or 0x88A8 VLAN tag at ingress, and pop the top VLAN tag at egress.
- Push 2 VLAN tags - push an outer Ethertype 0x8100 or 0x9100 (depends on the tunneling configuration), or 0x88A8 VLAN tag, and an inner Ethertype 0x8100 VLAN tags at ingress, and pop the top 2 VLAN tags at egress.
- Pop 1 VLAN tag - pop the top VLAN tag at ingress, and push an Ethertype 0x8100 VLAN tag or Ethertype 0x88A8 VLAN tag at egress.
- Pop 2 VLAN tags - pop the top 2 VLAN tags at ingress, and push an inner Ethertype 0x8100 VLAN tag and an outer Ethertype 0x8100 or 0x9100 (depends on the tunneling configuration), or 0x88A8 VLAN tag at egress.

Note This modification can only pop tags that are matched as part of the EFP.

- Rewrite 1 or 2 VLAN tags:
  - Rewrite outer tag
  - Rewrite outer 2 tags
  - Rewrite one outer tag and push an additional outer VLAN tag at ingress. Pop the outer VLAN tag and rewrite the second outer VLAN tag at egress.
  - Pop the outer VLAN tag and rewrite the second outer VLAN tag at ingress. Rewrite the outer VLAN tag and push an additional outer VLAN tag at egress.
- For each of the VLAN ID manipulations, you can specify Ethertype 0x88A8, 0x8100 or 0x9100 (depends on the tunneling configuration).

Valid Ingress Rewrite Actions

The following notations are used for the rewrite actions mentioned in the table:

- Translate 1-to-1 tag: Translates the outermost tag to another tag.
- Translate 1-to-2 tags: Translates the outermost tag to two tags.

- Translate 2-to-1 tags: Translates the outermost two tags to a single tag.
- Translate 2-to-2 tags: Translates the outermost two tags to two other tags.

Note Prior to Cisco IOS XR Software Release 7.8.1, the EtherType cannot be changed at 1-to-1 and 2-to-2 VLAN tag translation operations.

The following encapsulation types are supported:

- encapsulation dot1q <x>
- encapsulation dot1q priority-tagged (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- encapsulation dot1ad <x> (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- encapsulation dot1ad priority-tagged (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- encapsulation dot1ad <x> dot1q <y>
- encapsulation dot1q <x> second-dot1q <y> (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)
- dot1q tunneling ethertype 0x9100 (Supported from Cisco IOS XR Software Release 24.4.1 onwards)
- hw-module profile encap-exact (Supported from Cisco IOS XR Software Release 24.4.1 onwards)

The following lists the supported L2 sub-interface rewrite actions:

- rewrite ingress tag push dot1q <x> symmetric
- rewrite ingress tag push dot1ad <x> symmetric
- rewrite ingress tag push dot1ad <x> dot1q <y> symmetric
- rewrite ingress tag push dot1q <x> second-dot1q <y> symmetric (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)
- rewrite ingress tag pop 1 symmetric
- rewrite ingress tag pop 2 symmetric
- rewrite ingress tag translate 1-to-1 dot1q <x> symmetric
- rewrite ingress tag translate 1-to-1 dot1ad <x> symmetric
- rewrite ingress tag translate 1-to-2 dot1ad <x> dot1q <y> symmetric
- rewrite ingress tag translate 1-to-2 dot1q <x> second-dot1q <y> symmetric (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)
- rewrite ingress tag translate 2-to-1 dot1q <x> symmetric
- rewrite ingress tag translate 2-to-1 dot1ad <x> symmetric

- rewrite ingress tag translate 2-to-2 dot1ad <x> dot1q <y> symmetric
- rewrite ingress tag translate 2-to-2 dot1q <x> second-dot1q <y> symmetric (Supported on Q200-based line cards from Cisco IOS XR Software Release 24.1.1 onwards and supported for all systems in the Cisco 8000 Series Routers from Cisco IOS XR Software Release 24.4.1 onwards.)

Configure VLAN Header Rewrite

Configuring VLAN header rewrite involves:

- Creating a sub-interface
- Enabling L2 transport mode on the newly created sub-interface
- Defining the matching criteria (encapsulation mode) to be used in order to map single-tagged or double-tagged frames ingress on an interface to the appropriate service instance
- Specifying the encapsulation adjustment that is to be performed on the ingress frame

Configuration Example

```
/* Configuration without rewrite */

configure
 interface HundredGigE 0/0/0/0.1 l2transport
  encapsulation dot1q 10
 !
!

/* Configuration with rewrite */

/* PUSH 1 */
interface HundredGig0/0/0/0.1 l2transport
 encapsulation dot1q 10
 rewrite ingress tag push dot1q 20 symmteric
 !
!

/* POP 1 */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag pop 1 symmetric
 !
!

/* TRANSLATE 1-1 */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1q 10
  rewrite ingress tag translate 1-to-1 dot1q 20 symmetric
 !
!
!
```

Running Configuration (VLAN header rewrite on double-tagged sub-interface)

```
/* Configuration without rewrite */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
```

```
 !
!

/* Configuration with rewrite */

/* POP 2 */
interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
  rewrite ingress tag pop 2 symmetric
 !
!


/* TRANSLATE 1-2 */

interface HundredGigE0/0/0/0.1 l2transport
 encapsulation dot1ad 2 dot1q 1
  rewrite ingress tag translate 1-to-2 dot1ad 2 dot1q 1 symmetric
 !
!
```

Define Data-Forwarding Behaviour

The EFP can be used to designate the frames belonging to a particular Ethernet flow forwarded in the data path. These forwarding cases are supported for EFPs in Cisco IOS XR software:

- L2 Switched Service (Bridging)—The EFP is mapped to a bridge domain, where frames are switched based on their destination MAC address. This includes multipoint service:
  - Ethernet to Ethernet Bridging

Ethernet Flow Points Visibility

EFP Visibility feature enables you to configure multiple VLANs in the same bridge-domain.

An Ethernet flow point (EFP) service instance is a logical interface that connects a bridge domain to a physical port or to an EtherChannel group. One or two VLAN tags are used to identify the EFP.

Irrespective of number of ports available, you have flexibility to add more EFPs in one bridge group.

Configure Ethernet Flow Point

This example shows how to configure VLAN interfaces under a bridge domain with multiple EFPs.

Configuration Example

Create an EFP interface. To configure an EFP, use these interface configuration commands:

- **l2transport** - This command identifies a subinterface, a physical port, or a bundle-port parent interface as an EFP.
- **encapsulation** - This command is used in order to specify VLAN-matching criteria.
- **rewrite** - This command is used to specify the VLAN tag rewrite criteria.

```
/* Configure interfaces */
Router# configure
Router(config)# interface HundredGigE0/0/0/4.1 l2transport
Router(config-subif)# encapsulation dot1q 1
Router((config-subif))# rewrite ingress tag pop 1 symmetric
Router(config-subif))# exit
Router(config)# interface HundredGigE0/0/0/4.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router((config-subif))# rewrite ingress tag pop 1 symmetric
Router(config-subif))# exit
Router(config)# interface HundredGigE0/0/0/5.1 l2transport
Router(config-subif)# encapsulation dot1q 3
Router((config-subif))# rewrite ingress tag pop 1 symmetric
Router(config-subif))# exit
Router(config)# interface HundredGigE0/0/0/5.2 l2transport
Router(config-subif)# encapsulation dot1q 4
Router((config-subif))# rewrite ingress tag pop 1 symmetric
Router(config-subif))# exit

/* Configure a bridge bomain and interfaces to a bridge domain */
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/4.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/5.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit

Router(config-l2vpn)# bridge group bg2
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/4.2
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/5.2
Router(config-l2vpn-bg-bd-ac)# commit
```

Running Configuration

This section shows the EFP running configuration.

```
interface HundredGigE0/0/0/4.1 l2transport
encapsulation dot1q 1
rewrite ingress tag pop 1 symmetric
!
iinterface HundredGigE0/0/0/4.2 l2transport
encapsulation dot1q 12
rewrite ingress tag pop 1 symmetric
!
interface HundredGigE0/0/0/5.1 l2transport
encapsulation dot1q 3
rewrite ingress tag pop 1 symmetric
!
interface HundredGigE0/0/0/5.2 l2transport
encapsulation dot1q 4
rewrite ingress tag pop 1 symmetric
!
bridge group bg1
  bridge-domain bd1
   interface HundredGigE0/0/0/4.1
   !
```

```
   interface HundredGigE0/0/0/5.1
   !
  !
 !
 bridge group bg2
  bridge-domain bd2
   interface HundredGigE0/0/0/4.2
   !
   interface HundredGigE0/0/0/5.2
   !
  !
 !
```

Verification

Verify EFP configuration.

```
Router#show interfaces hundredGigE 0/0/0/4.2
Tue Sep 22 11:32:06.993 PDT
HundredGigE0/0/0/4.2 is up, line protocol is up
  Interface state transitions: 101
  Hardware is VLAN sub-interface(s), address is c4b2.39da.1620
  Layer 2 Transport Mode
  MTU 1518 bytes, BW 100000000 Kbit (Max: 100000000 Kbit)
     reliability Unknown, txload Unknown, rxload Unknown
  Encapsulation 802.1Q Virtual LAN,
    Outer Match: Dot1Q VLAN 2
    Ethertype Any, MAC Match src any, dest any
  loopback not set,
  Last link flapped 2d10h
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 3d18h
     21364536641 packets input, 2734660346522 bytes
     0 input drops, 0 queue drops, 0 input errors
     8420820982 packets output, 1077864630044 bytes
     0 output drops, 0 queue drops, 0 output errors

Router#show l2vpn bridge-domain summary
Tue Sep 22 11:31:29.819 PDT
Number of groups: 2, VLAN switches: 0
Number of bridge-domains: 510, Up: 510, Shutdown: 0, Partially-
programmed: 0
Default: 510, pbb-edge: 0, pbb-core: 0
Number of ACs: 1530 Up: 1275, Down: 255, Partially-programmed: 0
Number of PWs: 0 Up: 0, Down: 0, Standby: 0, Partially-programmed: 0
Number of P2MP PWs: 0, Up: 0, Down: 0, other-state: 0
Number of VNIs: 0, Up: 0, Down: 0, Unresolved: 0
```