# 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

***Table 1: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Double-Tagged 802.1ad Encapsulation Options for Layer 3 Physical and Bundle Subinterfaces | Release 7.3.2 | This feature enables you to increase the number of VLAN tags in an interface and an subinterface. You can enable this feature either on a physical interface or a bundle interface. When you configure this feature with the dual tag, interfaces check for IP addresses along with MAC addresses. **Verified Scalability Limits:**<br>• Total number of VLAN tags in an interface and a subinterface with single tag: 4094<br>• Total number of VLAN tags in an interface and a subinterface with double tag: 4094 * 4094 |

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.

VLAN ACs support the following modes of L2VPN operation:

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

**Note** The supported QinQ mode:

- EtherType 0x88A8 for outer VLAN tag
- EtherType 0x8100 for inner VLAN tag

### Double-Tagged 802.1ad Encapsulation Options for 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.

### Supported Encapsulation

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

| Interface Type | Encapsulation | Standard | Support Status |
|---|---|---|---|
| Layer 3 interface<br>Layer 3 subinterfaces<br>L3 bundle subinterface | Single-Tag Encapsulation | dot1ad | Not supported. |
| | | dot1q | Supported. |
| | Double-Tag Encapsulation | dot1ad <> dot1q<> | Supported. |
| | | dot1q <> dot1q<> | Not supported. |

# 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
```

### Restrictions

To configure VLAN sub-interface, the following restrictions are applicable.

- VLANs separated by comma are called a VLAN list. VLAN list is not supported.
- VLAN range is not supported. However, the router accepts and commits the configuration.
- If any sub-interface is already configured under a main interface, modifying the tunneling ether-type is not supported.

# 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

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. |
| Two outer most tags | The top tag must use EtherType 0x88A8.<br>The second top VLAN tag must use EtherType 0x8100. |

You can use wildcards while defining frames that map to a given EFP. EFPs can distinguish flows based on a single VLAN tag, a stack of VLAN tags or a combination of both (VLAN stack with wildcards). It provides the EFP model, a flexibility of being encapsulation agnostic, and allows it to be extensible as new tagging or tunneling schemes are added.

# 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 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 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 or 0x8100.

# 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** 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 dot1ad <x> dot1q <y>

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 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 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

# 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
```