L2VPN Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.3.x

First Published: 2021-02-01

Last Modified: 2021-09-01

Preface

Note   This product has reached end-of-life status. For more information, see the End-of-Life and End-of-Sale Notices.

This guide describes the Cisco 8000 Series Router configurations. The preface for the L2VPN Configuration Guide for Cisco 8000 Series Routers contains these sections:

- Changes to This Document, on page iii
- Obtaining Documentation and Submitting a Service Request, on page iii

Changes to This Document

This table lists the technical changes made to this document since it was first released.

Table 1: Changes to This Document

Date	Summary
October 2021	Republished for Release 7.3.2.
May 2021	Republished for Release 7.3.15.
February 2021	Initial release of this document.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the What's New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

New and Changed L2VPN Features

This table summarizes the new and changed feature information for the L2VPN Configuration Guide for Cisco 8000 Series Routers, and tells you where they are documented.

New and Changed L2VPN Features

Table 2: L2VPN Features Added or Modified in IOS XR Release 7.3.x

Feature	Description	Changed in Release	Where Documented
CFM on VPLS	This feature was introduced.	Release 7.3.2	CFM on VPLS, on page 47
Split-Horizon Groups	This feature was introduced.	Release 7.3.2	Split-Horizon Groups, on page 52
Traffic Storm Control	This feature was introduced.	Release 7.3.2	Traffic Storm Control, on page 54
Transparent Layer 2 Protocol Tunneling	This feature was introduced.	Release 7.3.2	Transparent Layer 2 Protocol Tunneling, on page 17
Virtual Private LAN Services	This feature was introduced.	Release 7.3.2	Virtual Private LAN Bridging Services , on page 37
GTP Load Balancing	This feature was introduced.	Release 7.3.2	GTP Load Balancing, on page 57
Inter-AS Mode for L2VPN Pseudowire	This feature was introduced.	Release 7.3.15	Inter-AS Mode, on page 72
Load Balance MPLS PW Traffic using Control-Word	This feature was introduced.	Release 7.3.15	Load Balance MPLS PW Traffic using Control-Word and Flow-Label, on page 79

Feature	Description	Changed in Release	Where Documented
Load Balance MPLS PW Traffic using Flow-Label	This feature was introduced.	Release 7.3.15	Load Balance MPLS PW Traffic using Control-Word and Flow-Label, on page 79
Pseudowire over MPLS	This feature was introduced.	Release 7.3.15	Pseudowire over MPLS , on page 61
Pseudowire VC Type 4	This feature was introduced.	Release 7.3.15	VLAN Mode, on page 67
Pseudowire VC Type 5	This feature was introduced.	Release 7.3.15	Ethernet Port Mode, on page 65
Support of Tagged or Untagged VLAN on Physical Rewrite	This feature was introduced.	Release 7.3.15	Configure Local Switching Between Attachment Circuits, on page 76

C H A P T E R 2

Introduction to Layer 2 Virtual Private Networks

This chapter provides the introduction to Layer 2 Virtual Private Networks.

Introduction to Layer 2 Virtual Private Networks

A Layer 2 Virtual Private Network (VPN) emulates a physical sub-network in an IP or MPLS network, by creating private connections between two points. Building an L2VPN network requires coordination between the service provider and the customer. The service provider establishes Layer 2 connectivity. The customer builds a network by using the data link resources obtained from the service provider. In an L2VPN service, the service provider does not require information about the customer's network topology and other information. This helps maintain customer privacy while using the service provider resources to establish the network.

The service provider requires Provider Edge (PE) routers with the following capabilities:

- Encapsulation of L2 protocol data units (PDU) into Layer 3 (L3) packets.

- Interconnection of any-to-any L2 transports.

- Support for MPLS tunneling mechanism.

- Process databases that include all information related to circuits and their connections.

L2VPN Interfaces Overview

An L2VPN network enables service providers (SPs) to provide L2 services to geographically disparate customer sites. Typically, an SP uses an access network to connect the customer to the core network. The connection between the customer site and the nearby SP edge router is known as an attachment circuit (AC). Traffic from the customer travels over this link to the edge of the SP core network. The traffic then tunnels through AC over the SP core network to another edge router. The edge router sends the traffic down another AC to the customer's remote site.

Benefits

- Allows SP to have a single infrastructure for both Layer 2 and Layer 3 services.

- Cost-effective due to converged IP or MPLS network.

CHAPTER 3

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 3: 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:** • Total number of VLAN tags in an interface and a subinterface with single tag: 4094 • 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.

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

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.

Supported Encapsulation

Table 4: 802.1ad Encapsulation Support for Layer 3 Interfaces and subinterfaces

Interface Type	Encapsulation	Standard	Support Status
Layer 3 interface Layer 3 subinterface Layer 3 bundle subinterface	Single-Tag Encapsulation	dot1ad	Not supported.
		dot1q	Supported.
	Double-Tag Encapsulation	dot1ad <> dot1q<>	Supported.
		dot1q <> dot1q<>	[1]Not supported.

[1] Layer 3 subinterface is supported only on Q200-based hardware platforms in the Cisco 8000 Series Routers.

Table 5: 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	Not supported.
		dot1q	Supported.
	Double-Tag Encapsulation	dot1ad <> dot1q<>	Supported.
		dot1q <> dot1q<>	[2]Supported.

[2] Only on Q200-based hardware platforms in the Cisco 8000 Series Routers.

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 restriction is applicable.

- VLANs separated by comma are called a VLAN list. VLAN list is not supported.

- VLANs separated by dashes are called a VLAN range. VLAN range 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 outer-most tag can use EtherType 0x8100 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 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 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>

- encapsulation dot1q <x> second-dot1q <y> (supported only on the Q200-based line cards in the Cisco 8000 Series Routers)

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 only on the Q200-based line cards in the Cisco 8000 Series Routers)

- 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 only on the Q200-based line cards in the Cisco 8000 Series Routers)

- 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 only on the Q200-based line cards in the Cisco 8000 Series Routers)

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
```

CHAPTER 4

Transparent Layer 2 Protocol Tunneling

This chapter introduces you to Transparent Layer 2 Protocol Tunneling to help initiate control packets from a local customer edge (CE) device to a remote CE device.

-

Transparent Layer 2 Protocol Tunneling

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
Transparent Layer 2 Protocol Tunneling	Release 7.3.2	This feature allows Layer 2 protocol data units (PDUs) to be kept intact and delivered across the service-provider network to the other side of the customer network. Such delivery is transparent because the VLAN and Layer 2 protocol configurations are maintained throughout.
		With this feature, service providers can send traffic from multiple customers across a core network without impacting the traffic of other customers.
		This feature is enabled by default.

This feature allows Layer 2 protocol data units (PDUs) to be tunneled across the core network without being interpreted and processed by intermediary network devices. Any packet on the L2 network is forwarded without any change. This feature is enabled by default.

You must configure the supported protocols only main and bundle interface. If you want a specific protocol packets to be punted over bundle members or subinterfaces, that protocol has to be enabled on the main interface as well.

When you want to use CFM and PVRST protocols, you must enable these protocols on a subinterface.

You can use this feature with the following protocols:

- Link Layer Discovery Protocol (LLDP)
- Cisco Discovery Protocol (CDP)
- Multiple Spanning Tree Protocol (MSTP)
- Per-VLAN Rapid Spanning Tree (PVRST)
- Connectivity Fault Management (CFM)
- Link Aggregation Control Protocol (LACP)
- Operation, Administration, Management (OAM)
- Synchronized Ethernet (SyncE)
- MAC security
- Priority Flow Control (PAUSE)

All packets on PW VPLS or VPWS are always tunnelled and no packet is sent to the CPU for processing (punted).

The following table depicts the behavior of the router when you enable a specific protocol on an interface.

L2 Protocol	Untagged Packet	Tagged Packet
Cisco Protocols	If Cisco protocols are enabled on the physical port, the traffic is sent to the CPU for processing. If Cisco protocols are disabled, the traffic is tunneled.	Traffic is always tunneled.
LLDP	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.
PVRST/PVRST+	If this protocol is enabled on the main port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	If this protocol is enabled on the subinterface, the traffic is sent to CPU for processing. If it is disabled, the traffic is tunneled.
MSTP	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.

L2 Protocol	Untagged Packet	Tagged Packet
CFM	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	If this protocol is enabled on the Xconnect, the traffic is sent to CPU for processing. If it is disabled, the traffic is tunneled.
LACP/SyncE/LOAM	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.
PFC	If this protocol is enabled on the physical port, the traffic is sent to the CPU for processing. If this protocol is disabled, the traffic is tunneled.	Traffic is always tunneled.

Configuration

You cannot disable transparent tunneling, this feature is enabled by default.

To display the protocols that are enabled per interface, use the **show ofa objects ethport base location 0/1/CPU0** command.

```
Router# show ofa objects ethport base location 0/1/CPU0
ethport element 0 (hdl:0x308f38e360):
  base
  |-- dpd_slf -   pending(cr/up/dl):0/0/0, sibling:0x3093b811c8, child:2, num_parents:3,
parent-trans_id:1523, visits:0
   color_mask:0, last_bwalk_id:0 num_bwalks_started:0
  |-- keylen - 4
  |-- trans_id - 489153
  |-- create_trans_id - 1523
  |-- obj_handle - 0x308f38e360
  |-- flag - 10
  |-- reason - 0
  |-- table_operation - 6
  |-- total_obj_size - 632
  |-- idempotent - 0
  |-- inflight - 0
  |-- table_prop - jid=169 mtime=(GMT)2021.Jan.09 13:05:46.670570
  |--  (cont'd)  - replayed=0times
  `-+ npu_results
    |-- npu0 - 0:Success
    |-- npu1 - 0:Success
    |-- npu2 - 0:Success
    `-- npu3 - 0:Success
  ofa_npu_mask_t npu_mask =>
…
  ofa_bool_t remote_chain_in_use => TRUE
  ofa_bool_t local_chain_in_use => TRUE
  uint8_t copc_profile => 0
  ofa_bool_t lldp_enable => FALSE
```

```
    ofa_bool_t slow_proto_enable => FALSE
    ofa_bool_t cdp_enable => (not set)
    ofa_bool_t pvrst_enable => FALSE
    ofa_bool_t mstp_enable => FALSE
    ofa_bool_t macsec_enable => FALSE
    ofa_bool_t mka_enable => FALSE
    ofa_bool_t pfc_enable => FALSE
    ofa_bool_t cfm_enable => FALSE
    dpa_npu_mask_t npu_bmap => (not set)


Router# show ofa objects l2if base location 0/1/CPU0
l2if element 0 (hdl:0x3094ba70a8):
  base
  |-- dpd_slf -   pending(cr/up/dl):0/0/0, sibling:0x308f8087c8, child:1, num_parents:1,
visits:0
   color_mask:0, last_bwalk_id:0 num_bwalks_started:0
  |-- flag - 10
      |-- flag.is_id_alloced - 0x1
  |-- keylen - 4
  |-- trans_id - 18311
  |-- create_trans_id - 18299
  |-- obj_handle - 0x3094ba70a8
  |-- obj_rc - 0x0
  |-- reason - 0
  |-- table_operation - 6
  |-- total_obj_size - 776
  |-- idempotent - 1
  |-- inflight - 0
  |-- table_prop - jid=137 mtime=(GMT)2021.Jun.21 14:53:56.644917
  |--  (cont'd)  - replayed=0times
  `-- obj_rc - 0:Success
  ofa_npu_mask_t npu_mask => 0 (not set)
  uint32_t member_count => 1
 @dpa_intf_t intf => 0x0f00000a
…
ofa_l2vpn_fwd_state_type fwd_state => (not set)
    ofa_bool_t cfm_enable => FALSE
    ofa_bool_t pvrst_enable => TRUE
    dpa_npu_mask_t npu_bmap => 1
```

To verify whether the L2 packet is flooded or forwarded by NPU, look at the interface counters. In case of flood, like multicast MAC, you will notice an increment in the output counters of the interface. When the traffic is forwarded with unicast MAC, you will notice an increment in the output counters only on the egress interface.

The following output displays the interface counters:

```
Router# show interface hundredGigE 0/0/2/0 accounting

HundredGigE0/0/2/0
  Protocol         Pkts In        Chars In      Pkts Out      Chars Out
  CDP                    0              0        163608        21923472
```

CHAPTER 5

Link Bundles for Layer 2 VPNs

An ethernet link bundle is a group of one or more ports that are aggregated together and treated as a single link. Each bundle has a single MAC, and a single configuration set (such as ACLs or QoS).

The advantages of link bundling are:

- Redundancy - Because bundles have multiple links, the failure of a single link does not cause a loss of connectivity.

- Increased bandwidth - On bundled interfaces traffic is forwarded over all available members of the bundle aggregating individual port capacity.

There are two types of link bundling supported depending on the type of interface forming the bundle:

- Ethernet interfaces

- VLAN interfaces (bundle sub-interfaces)

Interface Link Bundle

Cisco IOS XR software supports the EtherChannel method of forming bundles of Ethernet interfaces. EtherChannel is a Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.

IEEE 802.3ad encapsulation employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in an ethernet bundle are compatible. Links that are incompatible or have failed are automatically removed from the bundle.

Configuration Example

To create a link bundle between two routers, you must complete the following configurations:

1. Create a bundle instance

2. Map physical interface (s) to the bundle.

For an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

Configuration

```
/* Enter the global configuration mode and create the ethernet link bundle */
Router# configure
Router(config)# interface Bundle-Ether 3
Router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active links 1
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# exit

/* Map physical interfaces to the bundle */
/* Note: Mixed link bundle mode is supported only when active-standby operation is configured
 */
Router(config)# interface HundredGigE 0/0/0/0
Router(config-if)# bundle id 3 mode active
Router(config-if)# no shutdown
Router(config)# exit

Router(config)# interface HundredGigE 0/0/0/1
Router(config-if)# bundle id 3 mode active
Router(config-if)# no shutdown
Router(config-if)# exit

Router(config)# interface HundredGigE 0/0/0/2
Router(config-if)# bundle id 3 mode active
Router(config-if)# no shutdown
Router(config-if)# exit
```

Running Configuration

```
Router# show running-configuration
configure
interface Bundle-Ether 3
 ipv4 address 10.1.2.3 255.0.0.0
 bundle maximum-active links 32 hot-standby
 bundle minimum-active links 1
 bundle minimum-active bandwidth 30000000
!
interface HundredGigE 0/0/0/0
 bundle-id 3 mode active
!

interface HundredGigE 0/0/0/1
 bundle-id 3 mode active
!
interface HundredGigE 0/0/0/2
 bundle-id 3 mode active
!
```

Verification

Verify that interfaces forming the bundle are active and the status of the bundle is Up.

```
Router# show bundle bundle-ether 3
Tue Feb  4 18:24:25.313 UTC

Bundle-Ether1
  Status:                              Up
  Local links <active/standby/configured>:  3 / 0 / 3
  Local bandwidth <effective/available>:    30000000 (30000000) kbps
  MAC address (source):                1234.1234.1234 (Configured)
  Inter-chassis link:                  No
```

```
Minimum active links / bandwidth:        1 / 1 kbps
Maximum active links:                    32
Wait while timer:                        2000 ms
Load balancing:                          Default
LACP:                                    Operational
  Flap suppression timer:                Off
  Cisco extensions:                      Disabled
  Non-revertive:                         Disabled
mLACP:                                   Not configured
IPv4 BFD:                                Not configured

Port                 Device           State        Port ID          B/W, kbps
-------------------- ---------------  -----------  --------------   ----------
Hu0/0/0/0            Local            Active       0x8000, 0x0000   10000000
    Link is Active
Hu0/0/0/1            Local            Active       0x8000, 0x0000   10000000
    Link is Active
Hu0/0/0/2            Local            Active       0x8000, 0x0000   10000000
    Link is Active
```

VLAN Bundle

The procedure for creating VLAN bundle is the same as the procedure for creating VLAN sub-interfaces on a physical ethernet interface.

To configure VLAN bundles, complete the following configurations:

- Create a bundle instance.

- Create a VLAN interface (bundle sub-interface).

- Map the physical interface(s) to the bundle.

For a VLAN bundle to be active, you must perform the same configuration on both end points of the VLAN bundle.

Configuration

```
/* Enter global configuration mode and create VLAN bundle */
Router# configure
Router(config)# interface Bundle-Ether 2
Router(config-if)# ipv4 address 50.0.0.1/24
Router(config-if)# bundle maximum-active links 32 hot-standby
Router(config-if)# bundle minimum-active bandwidth 30000000
Router(config-if)# bundle minimum-active links 1
Router(config-if)# commit
Router(config-if)# exit

/* Create VLAN sub-interface and add to the bundle */
Router(config)# interface Bundle-Ether 2.201
Router(config-subif)# ipv4 address 12.22.1.1 255.255.255.0
Router(config-subif)# encapsulation dot1q 201
Router(config-subif)# commit
Router(config-if)# exit

/* Map the physical interface to the bundle */
Router(config)# interface HundredGigE 0/0/0/14
Router(config-if)# bundle id 2 mode on
Router(config-if)# no shutdown
Router(config-if)# commit
Router(config-if)# exit
```

```
/* Repeat the above steps for all the member interfaces:
   0/0/0/15, 0/0/0/16 and 0/0/0/17 in this example */
```

Running Configuration

```
Router# show running-configuration
configure
interface Bundle-Ether2
 ipv4 address 50.0.0.1 255.255.255.0
 mac-address 1212.1212.1212
 bundle maximum-active links 32 hot-standby
 bundle minimum-active links 1
 bundle minimum-active bandwidth 30000000
!
interface Bundle-Ether2.201
 ipv4 address 12.22.1.1 255.255.255.0
 encapsulation dot1q 201
!
interface HundredGigE0/0/0/14
 bundle id 2 mode on
!
interface HundredGigE0/0/0/15
 bundle id 2 mode on
!
interface HundredGigE0/0/0/16
 bundle id 2 mode on
!
interface HundredGigE0/0/0/17
 bundle id 2 mode on
!
```

Verification

Verify that the VLAN status is UP.

```
Router# show interfaces bundle-ether 2.201

Bundle-Ether2.201 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is 28c7.ce01.dc7b
  Internet address is 12.22.1.1/24
  MTU 1518 bytes, BW 20000000 Kbit (Max: 20000000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 201,  loopback not set,
  Last link flapped 07:45:25
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, 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
     2938 packets input, 311262 bytes, 0 total input drops
  - - -
  - - -
```

References for Configuring Link Bundles

Characteristics of Link Bundles

- Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).

- Physical layer and link layer configuration are performed on individual member links of a bundle.

- Configuration of network layer protocols and higher layer applications is performed on the bundle itself.

- A bundle can be administratively enabled or disabled.

- Each individual link within a bundle can be administratively enabled or disabled.

- Ethernet link bundles are created in the same way as Etherokinet channels, where the user enters the same configuration on both end systems.

- The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.

- When LACP configured, each link within a bundle can be configured to allow different keepalive periods on different members.

- Load balancing is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.

- QoS is supported and is applied proportionally on each bundle member.

- Link layer protocols, such as CDP, work independently on each link within a bundle.

- Upper layer protocols, such as routing updates and hello messages, are sent over any member link of an interface bundle.

- Bundled interfaces are point to point.

- A link must be in the UP state before it can be in distributing state in a bundle.

- Access Control List (ACL) configuration on link bundles is identical to ACL configuration on regular interfaces.

- Multicast traffic is load balanced over the members of a bundle. For a given flow, internal processes select the member link and all traffic for that flow is sent over that member.

Cisco IOS-XR software supports the following methods of forming bundles of Ethernet interfaces:

Cisco IOS-XR software supports the following methods of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.

  For each link configured as bundle member, information is exchanged between the systems that host each end of the link bundle:

  - A globally unique local system identifier

  - An identifier (operational key) for the bundle of which the link is a member

- An identifier (port ID) for the link

- The current aggregation status of the link

This information is used to form the link aggregation group identifier (LAG ID). Links that share a common LAG ID can be aggregated. Individual links have unique LAG IDs.

The system identifier distinguishes one router from another, and its uniqueness is guaranteed through the use of a MAC address from the system. The bundle and link identifiers have significance only to the router assigning them, which must guarantee that no two links have the same identifier, and that no two bundles have the same identifier.

The information from the peer system is combined with the information from the local system to determine the compatibility of the links configured to be members of a bundle.

Bundle MAC addresses in the routers come from a set of reserved MAC addresses in the backplane. This MAC address stays with the bundle as long as the bundle interface exists. The bundle uses this MAC address until the user configures a different MAC address. The bundle MAC address is used by all member links when passing bundle traffic. Any unicast or multicast addresses set on the bundle are also set on all the member links.

- EtherChannel—Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.

Link Aggregation Through LACP

The optional Link Aggregation Control Protocol (LACP) is defined in the IEEE 802 standard. LACP communicates between two directly connected systems (or peers) to verify the compatibility of bundle members. For a router, the peer can be either another router or a switch. LACP monitors the operational state of link bundles to ensure these:

- All links terminate on the same two systems.

- Both systems consider the links to be part of the same bundle.

- All links have the appropriate settings on the peer.

LACP transmits frames containing the local port state and the local view of the partner system's state. These frames are analyzed to ensure both systems are in agreement.

C H A P T E R 6

Layer 2 Bridging Services

This module provides the conceptual and configuration information for Layer 2 Bridging Services.

Layer 2 Bridging

You can use Layer 2 bridging services in data centers, campuses, and global networks.

A logical bridge contains the following components:

Bridge Domain

The bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports. Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain.

A learned MAC address has an age attribute. MAC address is remembered for a specified aging time and is forgotten if it has not been seen in received traffic for a age period.

A switch assigns a local significant ID to each bridge domain, which is known as the bridge domain ID. Many legacy switches use VLAN as bridge domain ID, which is known as bridging VLAN.

Bridge Port

A logical bridge port identifies a unique network segment in a bridge domain. L2 traffic transits a bridge domain through logical bridge ports. A logical bridge port is independent of the encapsulation of L2 traffic

such as VLAN or MPLS. A bridge port performs native bridging functions, such as forwarding, destination MAC address lookup, source MAC address learning, and aging.

MAC Address Table

Forwarding or filtering information table is also known as MAC address table. Each bridge domain has a unique MAC address table. The table consists of MAC address entries. When an Ethernet frame is received on a bridge port, the source MAC address and bridge port are recorded in the MAC address table. This information is used for traffic forwarding in reverse direction.

The following is an example of a MAC address table:

MAC Address Table	
MAC Address	**Ports**
1001.1001.2002	Port 2
1001.1001.2003	Port 5
1001.1001.2004	Drop **Note** Drop is not supported in this release.

Replication Member List

A replication member list is a list of virtual bridge ports that allow traffic flooding. A bridge domain has one replication list per each bridge domain.

Configure a Bridge Domain

Perform the following tasks to configure a bridge domain:

Create a Bridge Domain

Perform this task to create a bridge domain.

Configuration Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# commit
```

Running Configuration

This section shows the bridge domain running configuration.

```
configure
 l2vpn
  bridge group bg1
```

```
     bridge-domain bd1
    !
   !
```

Associate Members with a Bridge Domain

After a bridge domain is created, perform this task to assign interfaces to the bridge domain.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# commit
```

Running Configuration

This section shows the running configuration.

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface HundredGigE0/0/0/0
   !
  !
```

Configure Bridge Domain Parameter

To configure bridge domain parameter, associate this parameter with a bridge domain:

  • Flooding—Flooding is enabled by default.

Configuration Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# flooding disable
Router (config-l2vpn-bg-bd)# commit
```

Running Configuration

This section shows the bridge domain parameters running configuration.

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding disable
    !
  !
```

Disable a Bridge Domain

Perform this task to disable a bridge domain. When a bridge domain is disabled, all ACs that are associated with the bridge domain are disabled. You are still able to attach or detach members to the bridge domain and the ACs that are associated with the bridge domain.

Configuration Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# shutdown
Router (config-l2vpn-bg-bd)# commit
```

Running Configuration

This section shows the running configuration.

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    shutdown
   !
  !
```

VLAN Bridging

VLAN bridging is the simplest mode of L2 bridging. In this mode, all traffic that is received on the switch is either Ethernet II frames or IEEE 802.3 frames.

In modern networks, a majority of the Ethernet frames are in Ethernet II frame format. Legacy L2 protocol traffic, such as spanning tree protocol and CDP are in IEEE 802.3 frame format.

Topology

This topology shows a VLAN bridging in a campus network. Each L2 flood domain extends over different floors in the same building, and also other buildings. MAC hosts move freely between office buildings without dropping TCP and IP sessions. The advantage of host mobility is that VLAN bridging is used instead of IP segmentation (subnet routing).

Figure 1: VLAN Bridging
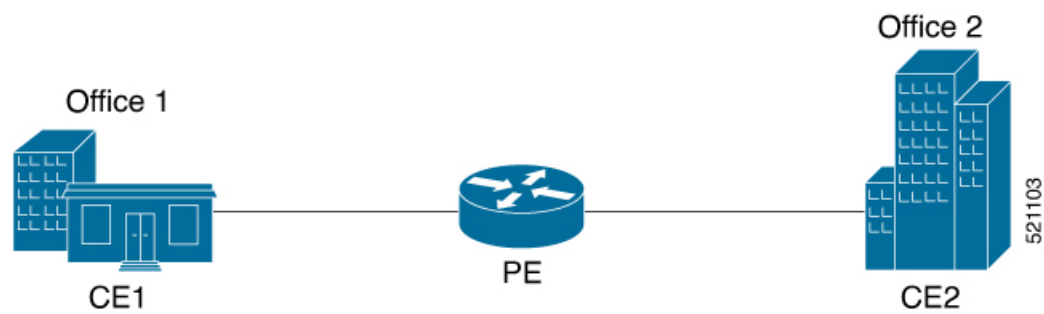
The router at the edge of a core in the network aggregates L2 traffic from local buildings, which are also known as customer edge (CE) devices. The ingress traffic from CE on the router is tagged with either single or double VLAN. The router classifies ingress traffic to different L2 bridge domains and performs optional VLAN tag rewrite. At the egress, the router sends the traffic to a different CE or to a remote router. On the remote router, the traffic is bridged to local office buildings after optional VLAN tag rewrite.

Configure VLAN Bridging

Perform this task to configure VLAN bridging.

```
/* Configure Attachment Circuits (ACs) */
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 associate ACs 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 VLAN bridging 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 VLAN bridging 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


Router#show l2vpn forwarding bridge-domain location 0/RP0/CPU0
```

```
Tue Sep 22 11:36:01.888 PDT
                                      Bridge       MAC
Bridge-Domain Name                    ID     Ports HW addr SW addr Flooding Learning State
------------------------------ ------ ----- ------- ------- -------- -------- ---------
bg1:bd1                               511    2     405     405     Enabled  Enabled  UP
bg1:bd2                               510    2     405     405     Enabled  Enabled  UP
-----------------------------------------------------------------------------------------

Router#show l2vpn forwarding bridge-domain bg1:bd1 location 0/RP0/CPU0
Tue Sep 22 11:36:37.141 PDT
                                      Bridge       MAC
Bridge-Domain Name                    ID     Ports HW addr SW addr Flooding Learning State
------------------------------ ------ ----- ------- ------- -------- -------- ---------
bg1:bd1                               511    2     405     405     Enabled  Enabled  UP
-----------------------------------------------------------------------------------------
```

MAC Address-related Parameters

The MAC address table contains a list of known MAC addresses and their forwarding information. The MAC address table is managed and stored on the route processor (RP) card.

These topics provide information about the MAC address-related parameters:

MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To perform flooding within the broadcast domain, all unknown unicast, broadcast, and multicast addresses are flooded to all attachment circuits. Therefore, a provider edge (PE) device replicates packet across the attachment circuits.

MAC Address-based Forwarding

To forward a frame, a PE must associate a destination MAC address with an attachment circuit. This type of association is provided through a static configuration on each PE or through dynamic learning.

MAC Address Source-based Learning

When a frame arrives on a bridge port and the source MAC address is unknown to the receiving PE router, the source MAC address is associated with the attachment circuit. Outbound frames of the MAC address are forwarded to the appropriate attachment circuit.

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. During the learning process, the data plane hardware notifies control plane about the source MAC address and its associated bridge port. Control plane keeps a note of it on RP and programs the MAC address and its bridge port to MAC tables on all forwarding ASIC in the system.

Note   You can set a MAC address on an AC in a bridge domain. This MAC address is statically programmed on the MAC table. This MAC address can neither age nor move to another AC in the bridge domain through dynamic learning. For example, if a static MAC address is configured on AC1 (port 1) and then, if you send a packet with the same MAC address as source MAC address on AC2 (port 2), then you cannot attach this MAC address to AC2 as a dynamic MAC address. Therefore, do not send any packet with the MAC address which is the same static MAC address configured.

MAC Address Aging

A MAC address in the MAC table is considered valid only for the duration of the MAC address aging time. When the time expires, the relevant MAC entries are removed. When the MAC aging time is configured only under a bridge domain, all the attachment circuits in the bridge domain use that configured MAC aging time.

A bridge forwards, floods, or drops packets based on the bridge table. The bridge table maintains both static entries and dynamic entries. Static entries are entered by the network manager or by the bridge itself. Dynamic entries are entered by the bridge learning process. A dynamic entry is automatically removed after a specified length of time, known as *aging time*, from the time the entry was created or last updated.

If hosts on a bridged network are likely to move, decrease the aging-time to enable the bridge to adapt to the change quickly. If hosts do not transmit continuously, increase the aging time to record the dynamic entries for a longer time, thus reducing the possibility of flooding when the hosts transmit again.

The range of MAC address aging time is from 300 seconds to 30,000 seconds. The maximum MAC address aging time among all bridges is considered for calculating the age. You cannot configure the MAC address aging time on each AC interface. Configure MAC address aging time in the bridge domain configuration mode. There is no show command to display the highest MAC address aging time.

Note When you configure the different aging time for each bridge domains, the system considers the highest value of all the bridge domains. For example, if you configure the aging time on bd1 as 300 seconds, on bd2 as 600 seconds, and bd3 as 800 seconds, MAC address aging time is taken as 800 seconds for all the bridge domains bd1, bd2, and bd3. All the three bridge domains age out at 800 seconds.

MAC Address Limit

The MAC address limit is used to alert the user when MAC addresses in a bridge domain exceed a certain threshold. The maximum MAC address limit is 131072.

When a limit is exceeded, the system displays the following notifications:

- Syslog (default)

- Simple Network Management Protocol (SNMP) trap

- Syslog and SNMP trap

- None (no notification)

To generate syslogs messages and SNMP trap notifications, use the **mac limit notification both** command in the L2VPN bridge-domain configuration mode.

MAC address limit action applies only when the number of local MAC addresses exceeds the configured limit. When the MAC limit threshold is not configured, the default MAC address limit is 131072.

Configure MAC-related Parameters

These tasks describe how to configure the MAC address-related parameters:

Configure the MAC Address Source-based Learning

MAC address source-based learning is enabled by default, Perform this task to disable the MAC address source-based learning.

Configuration Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# mac
Router (config-l2vpn-bg-bd-mac)# learning disable
Router (config-l2vpn-bg-bd-mac)# commit
```

Running Configuration

This section shows the MAC address source-based learning running configuration.

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    mac
     learning disable
     !
  !
```

Configure the MAC Address Limit

Perform this task to configure the parameters for the MAC address limit.

Note   You cannot set the custom value for the MAC address limit. You can configure the MAC address limit only to a maximum value, which is 131072.

Configuration Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# mac
Router (config-l2vpn-bg-bd-mac)# limit
Router (config-l2vpn-bg-bd-mac-limit)# maximum 131072
Router (config-l2vpn-bg-bd-mac-limit)# notification both
Router (config-l2vpn-bg-bd-mac-limit)# exit
Router (config-l2vpn-bg-bd)# exit
Router (config-l2vpn-bg-bd)# mac limit threshold 80
Router (config-l2vpn-bg-bd-mac-limit)# commit
```

Running Configuration

This section shows the MAC address limit running configuration.

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    mac
     limit
     maximum 131072
     notification both
    !
    mac limit threshold 80
   !
  !
```

Configure the MAC Address Aging

Perform this task to configure the parameters for MAC address aging.

Configuration Example

```
Router# configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group bg1
Router (config-l2vpn-bg)# bridge-domain bd1
Router (config-l2vpn-bg-bd)# mac
Router (config-l2vpn-bg-bd-mac)# aging
Router (config-l2vpn-bg-bd-mac-aging)# time 300
Router (config-l2vpn-bg-bd-mac-aging)# commit
```

Running Configuration

This section shows the MAC address aging running configuration.

```
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    mac
     aging
      time 300
     !
    !
```

Flooding Disable

The Flooding Disable feature prevents forwarding of Broadcast, Unknown-unicast and Multicast (BUM) traffic on the bridge domain. You can disable flooding of BUM traffic at the bridge level. By disabling flooding at the bridge level, you can prevent forwarding of BUM traffic on AC).

You can also disable only unknown unicast traffic at the bridge level. By disabling flooding of unknown unicast traffic at the bridge level, you can prevent forwarding of unknown unicast traffic on AC.

Configure Flooding Disable

Perform this task to configure Flooding Disable feature.

You can disable flooding of:

- BUM traffic at the bridge level

- Unknown-unicast traffic at the bridge level

However, the flooding disable of unknown-unicast traffic at the bridge level takes effect only when the **flooding disable** command is not configured for BUM traffic at the bridge level.

Configuration Example

```
/* Configuration to disable flooding of BUM traffic at the bridge level */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#flooding disable
Router(config-l2vpn-bg-bd)#commit

/* Configuration to disable flooding of unknown-unicast traffic at the bridge level */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#flooding unknown-unicast disable
Router(config-l2vpn-bg-bd)#commit
```

Running Configuration

This section shows flooding disable running configuration.

```
/* Configuration to disable flooding of BUM traffic at the bridge level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding disable
   !

/* Configuration to disable flooding of unknown-unicast traffic at the bridge level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding unknown-unicast disable
   !
  !
```

Virtual Private LAN Bridging Services

Table 7: Feature History Table

Feature Name	Release Information	Feature Description

Virtual Private LAN Bridging Services	Release 7.3.2	This feature employs PE routers connected by a mesh of tunnels, enabling you to connect multiple customer devices in a single bridged domain. Such a setup allows service providers to seamlessly offer a variety of services that they can provision rapidly.

A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

VPLS is a multipoint Layer 2 VPN technology that connects two or more provider edge (PE) devices using bridging. A bridge domain, which is the building block for multipoint bridging, is present on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment circuits. The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

Now, the service provider network starts switching the packets within the bridged domain specific to the customer by looking at destination MAC addresses. All traffic with unknown, broadcast, and multicast destination MAC addresses is flooded to all the connected customer edge devices, which connect to the service provider network. The network-facing provider edge devices learn the source MAC addresses as the packets are flooded. The traffic is unicasted to the customer edge device for all the learned MAC addresses.

VPLS Architecture

The VPLS architecture allows for the end-to-end connection between the PE routers to provide multipoint ethernet services. Following figure shows a VPLS architecture illustrating the interconnection between the network provider edge (N-PE) nodes over an IP/MPLS network.

Figure 2: VPLS Architecture
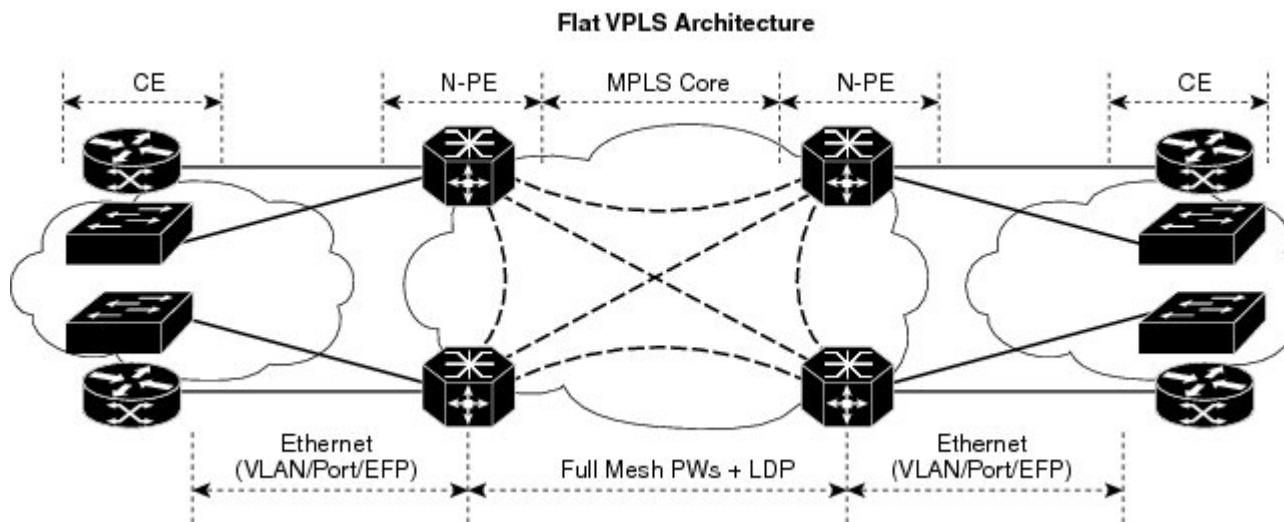
The VPLS network requires the creation of a bridge domain (Layer 2 broadcast domain) on each of the PE routers. The VPLS provider edge device holds all the VPLS forwarding MAC tables and bridge domain information. In addition, it is responsible for all flooding broadcast frames and multicast replications.

The PEs in the VPLS architecture are connected with a full mesh of Pseudowires (PWs). A Virtual Forwarding Instance (VFI) is used to interconnect the mesh of pseudowires. A bridge domain is connected to a VFI to

create a Virtual Switching Instance (VSI), that provides Ethernet multipoint bridging over a PW mesh. VPLS network links the VSIs using the MPLS pseudowires to create an emulated Ethernet Switch.

With VPLS, all customer equipment (CE) devices participating in a single VPLS instance appear to be on the same LAN and, therefore, can communicate directly with one another in a multipoint topology, without requiring a full mesh of point-to-point circuits at the CE device.

VPLS transports Ethernet IEEE 802.3, VLAN IEEE 802.1q, and VLAN-in-VLAN (q-in-q) traffic across multiple sites that belong to the same Layer 2 broadcast domain. VPLS offers simple VLAN services that include flooding broadcast, multicast, and unknown unicast frames that are received on a bridge. The VPLS solution requires a full mesh of pseudowires that are established among PE routers. The VPLS implementation is based on Label Distribution Protocol (LDP)-based pseudowire signaling.

Pseudowires

A pseudowire (PW) is a point-to-point connection between pairs of PE routers. Its primary function is to emulate services like Ethernet over an underlying core MPLS network through encapsulation into a common MPLS format. By encapsulating services into a common MPLS format, a pseudowire allows carriers to converge their services to an MPLS network.

Note Multiple PWs within a bridge domain are supported. However, multiple PWs to the same destination within the same bridge domain are not supported.

Perform this task to configure a pseudowire under a bridge domain.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-pw)# commit
```

Running Configuration

This section shows the pseudowire running configuration.

```
l2vpn
 bridge group BG1
  bridge-domain BD1
   neighbor 10.0.0.2 pw-id 1
  !
!
```

Pseudowire Statistics

- Use the **hw-module profile l2fib pw-stats** command to enable PW statistics. After you enable PW statistics, you must reload all slots for configuration to take effect.

- Pseudowire statistics are not supported on line cards based on Q100 Silicon.

- Only aggregate and unicast statistics are supported for ingress PW (PW disposition). Multicast, broadcast, and unknown unicast statistics are not supported for PW.

- Statistics resource is limited, and enabling statistics on PW impacts forwarding performance; therefore, you must enable PW statistics manually using the **hw-module profile** command.

The following example shows the pseudowire statistics before and after the reload:

```
Before the reload:
Router# show hw-module profile l2fib
-----------------------------------------------------------
Knob                            Status      Applied   Action
-----------------------------------------------------------
PW-Stats                        Configured  No        Reload /*configured but not
reloaded/applied yet*/
BD-Flush-Convergence            Unconfigured N/A      None

After the reload:
Router# show hw-module profile l2fib
-----------------------------------------------------------
Knob                            Status      Applied   Action
-----------------------------------------------------------
PW-Stats                        Configured  Yes       None /*configured but not
reloaded/applied yet*/
BD-Flush-Convergence            Unconfigured N/A      None
```

The following output shows the aggregate and unicast statistics for ingress PW:

```
Router# show l2vpn forwarding bridge-domain detail location 0/RP0/CPU0 | inc "state:Nbor|XC
 ID|received"
    XC ID: 0x1
     packets: received 2081 (multicast 0, broadcast 2081, unknown unicast 0, unicast 0),
sent 998
     bytes: received 266368 (multicast 0, broadcast 266368, unknown unicast 0, unicast 0),
 sent 127744
    XC ID: 0x2
     packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent
3079
     bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent
394112
    XC ID: 0xa0000001
     packets: received 998 (unicast 0), sent 1996
     bytes: received 145708 (unicast 0), sent 307384
```

Virtual Forwarding Instance

VPLS is based on the characteristic of virtual forwarding instance (VFI). A VFI is a virtual bridge port that is capable of performing native bridging functions, such as forwarding, based on the destination MAC address, source MAC address learning and aging, and so forth.

A VFI is created on the PE router for each VPLS instance. The PE routers make packet-forwarding decisions by looking up the VFI of a particular VPLS instance. The VFI acts like a virtual bridge for a given VPLS instance. More than one attachment circuit belonging to a given VPLS are connected to the VFI. The PE router establishes emulated VCs to all the other PE routers in that VPLS instance and attaches these emulated VCs to the VFI. Packet forwarding decisions are based on the data structures maintained in the VFI.

Configure a VFI

Perform this task to confiure a Virtual Forwarding Instance (VFI) on all provider edge devices under the bridge domain and associate a pseudowire with the VFI.

```
Router# configure
```

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# vfi V1
Router(config-l2vpn-bg-bd-vfi)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

Running Configuration

This section shows the VFI running configuration.

```
l2vpn
 bridge group BG1
  bridge-domain BD1
   vfi V1
    neighbor 10.0.0.2 pw-id 1
  !
!
```

VPLS PW Transport Types

The following are the supported transport types for VPLS PW:

- LDP

- Segment Routing

- LDP over TE

- BGP-LU for inter-AS C topology

Pseudowire Types and Transported Tags

The following pseudowire types are supported:

- Ethernet Port Mode—Ethernet port mode is supported for pseudowire over MPLS. The virtual connection (VC) type 5 is known as an Ethernet port-based PW. In this mode, both ends of a pseudowire are connected to Ethernet ports and allow a complete ethernet trunk to be transported. The ingress PE transports frames received on a main interface or subinterface. This feature nullifies the need for a dummy tag and reduces overhead. In addition, frame tagging is no longer necessary.

- VLAN Mode—VLAN mode is supported for pseudowire over MPLS. A virtual connection (VC) type 4 is the VLAN-based PW. The ingress PE does not remove the incoming VLAN tags that are to be transported over the PW. VC type 4 inserts an extra dummy tag with VLAN 0 onto the frame which is removed on the other side. This mode helps the service provider to segregate traffic for each customer based on the VLAN.

  In this mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4. VLAN-based (VC Type 4) pseudowires ensure a VLAN tag is transported over the pseudowire by pushing a dummy tag at the attachment circuit ingress. If the rewrite rule pushes two or more tags, a dummy tag is not needed because these VLAN tags are transported over the pseudowire. On the remote router, the dummy tag, if added, is removed before egress.

Note   A mix of type 4 and type 5 PWs under the same VFI is not supported. All PWs must be of the same type.

Pseudowire Control-Word

A control-word is an optional 4-byte field located between the MPLS label stack and the Layer 2 payload in the pseudowire packet. The control word carries generic and Layer 2 payload-specific information.

The control-word has the following functions:

- Pad small packets—If the AToM packet does not meet the minimum length then the frame is padded to meet the minimum length on the ethernet link. Because the MPLS header has no length that indicates the length of the frames, the control word holds a length field indicating the length of the frame. If the received AToM packet in the egress PE router has a control word with a length that is not 0, the router knows that padding was added and can correctly remove the padding before forwarding the frames.

- Carry control bits of the Layer 2 header of the transported protocol.

- Preserve the sequence of the transported frames—Preserve the sequence of the transported frames—The first packet sent onto the PW has a sequence number of 1 and increments for each subsequent packet by one until it reaches 65535. If such out of sequence packets are detected, the packets are dropped. The re-ordering for the out-of-sequence packet is not performed.

- Load balancing—Load balancing allows the router to correctly identify the Ethernet PW packet over an IP packet, thus preventing the selection of wrong equal-cost multipath (ECMP) path for the packet that leads to the misordering of packets. The **control-word** keyword is inserted immediately after the MPLS label to separate the payload from the MPLS label over a PW. The control word carries layer 2 control bits and enables sequencing.

- Facilitate fragmentation and reassembly—This is used to indicate the payload fragmentation:

  - 00 = unfragmented

  - 01 = 1st fragment

  - 10 = last fragment

  - 11 = intermediate fragment

Configure Pseudowire Control-Word

Perform this task to configure the pseudowire control-word.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class path1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-pw)# pw-class path1
Router(config-l2vpn-bg-bd-pw)# commit
```

Running Configuration

This section shows the pseudowire control-word running configuration.

```
l2vpn
 pw-class path1
```

```
  encapsulation mpls
   control-word
 bridge group BG1
  bridge-domain BD1
   neighbor 10.0.0.2 pw-id 1
    pw-class path1
  !
!
```

Load Balancing using Flow Aware Transport Pseudowire (FAT PW)

Routers typically load balance traffic based on the lowermost label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric load balancing. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) to a destination PE.

Flow Aware Transport Pseudowires (FAT PW) provides the capability to identify individual flows within a pseudowire and provide routers the ability to use these flows to load balance traffic. FAT PWs are used to load balance traffic in the core when equal-cost multipaths (ECMP) are used. A flow label is created based on individual packet flows entering a pseudowire; and is inserted as the lowermost label in the packet. Routers can use the flow label for load balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle-links.

Figure 3: FAT PW with two flows distributing over ECMPs and Bundle-Links
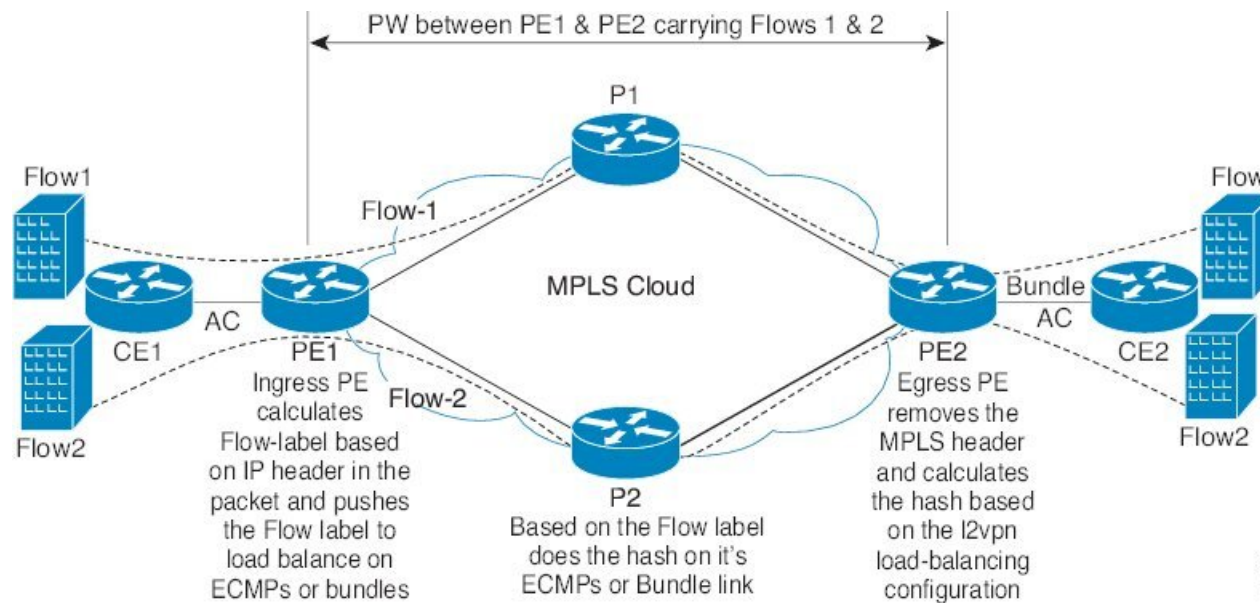
An additional label is added to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of the label stack (EOS) bit set and inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it's possible to distribute flows over ECMPs and link bundles.

You cannot send MPLS OAM ping traffic over a FAT PW, since there is no flow label support for MPLS OAM.

Configure load balancing using FAT PW

Perform this task to enable load balancing with ECMP and FAT PW. Creating a PW class in L2VPN configuration leads to load balancing.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class FLOW
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# flow-label both
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# neighbor 10.0.0.2 pw-id 1
Router(config-l2vpn-bg-bd-pw)# pw-class FLOW
Router(config-l2vpn-bg-bd-pw)# commit
```

Running Configuration

This section shows the running configuration.

```
l2vpn
 pw-class FLOW
  encapsulation mpls
   flow-label both
 bridge group BG1
  bridge-domain BD1
   neighbor 10.0.0.2 pw-id 1
    pw-class FLOW
  !
!
```

VPLS Discovery and Signaling

VPLS is a Layer 2 multipoint service and it emulates LAN service across a WAN service. VPLS enables service providers to interconnect several LAN segments over a packet-switched network and make it behave as one single LAN. Service provider can provide a native Ethernet access connection to customers using VPLS.

The VPLS control plane consists of two important components, autodiscovery and signaling:

- VPLS Autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS Autodiscovery enables each VPLS PE router to discover the other provider edge (PE) routers that are part of the same VPLS domain.

- Once the PEs are discovered, pseudowires (PWs) are signaled and established across each pair of PE routers forming a full mesh of PWs across PE routers in a VPLS domain.

Figure 4: VPLS Autodiscovery and Signaling

L2-VPN	Multipoint	
Discovery	BGP	
Signaling Protocol	LDP	BGP
Tunneling Protocol	MPLS	

BGP-based VPLS Autodiscovery

An important aspect of VPN technologies, including VPLS, is the ability of network devices to automatically signal to other devices about an association with a particular VPN. Autodiscovery requires this information to be distributed to all members of a VPN. VPLS is a multipoint mechanism for which BGP is well suited.

BGP-based VPLS autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS autodiscovery enables each VPLS PE router to discover the other provider edge (PE) routers that are part of the same VPLS domain. VPLS Autodiscovery also tracks when PE routers are added to or removed from the VPLS domain. When the discovery process is complete, each PE router has the information required to setup VPLS pseudowires (PWs).

Even when BGP autodiscovery is enabled, pseudowires can be manually configured for VPLS PE routers that are not participating in the autodiscovery process.

BGP Auto Discovery With BGP Signaling

The implementation of VPLS in a network requires the establishment of a full mesh of PWs between the provider edge (PE) routers. The PWs can be signaled using BGP signaling.

Figure 5: Discovery and Signaling Attributes
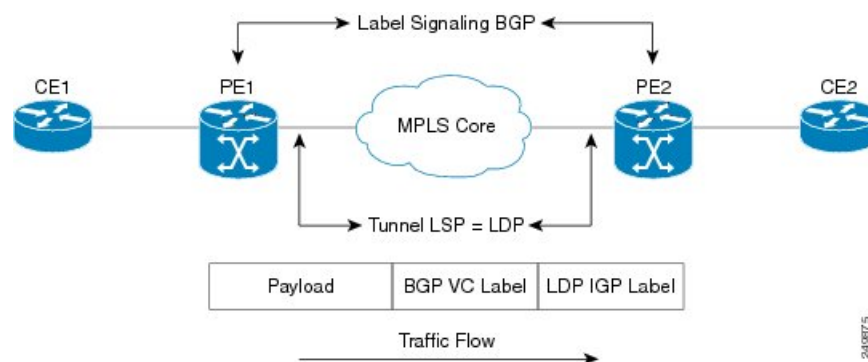
The BGP signaling and autodiscovery scheme has the following components:

• A means for a PE to learn which remote PEs are members of a given VPLS. This process is known as autodiscovery.

• A means for a PE to learn the pseudowire label expected by a given remote PE for a given VPLS. This process is known as signaling.

The BGP Network Layer Reachability Information (NLRI) takes care of the above two components simultaneously. The NLRI generated by a given PE contains the necessary information required by any other

PE. These components enable the automatic setting up of a full mesh of pseudowires for each VPLS without having to manually configure those pseudowires on each PE.

NLRI Format for VPLS with BGP AD and Signaling

The following figure shows the NLRI format for VPLS with BGP AD and Signaling

Figure 6: NLRI Format

Length (2 octets)
Route Distinguisher (8 octets)
VE ID (2 octets)
VE Block Offset (2 octets)
VE Block Size (2 octets)
Label Base (3 octets)

BGP Auto Discovery With LDP Signaling

Signaling of pseudowires requires exchange of information between two endpoints. Label Distribution Protocol (LDP) is better suited for point-to-point signaling. The signaling of pseudowires between provider edge devices, uses targeted LDP sessions to exchange label values and attributes and to configure the pseudowires.

Figure 7: Discovery and Signaling Attributes
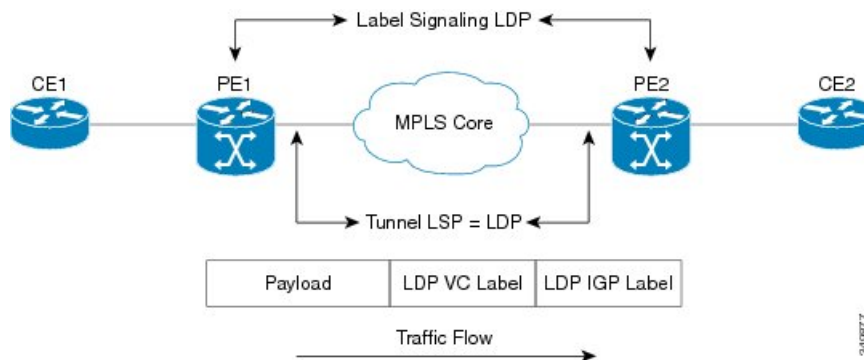
A PE router advertises an identifier through BGP for each VPLS. This identifier is unique within the VPLS instance and acts like a VPLS ID. The identifier enables the PE router receiving the BGP advertisement to identify the VPLS associated with the advertisement and import it to the correct VPLS instance. In this manner, for each VPLS, a PE router learns the other PE routers that are members of the VPLS.

The LDP protocol is used to configure a pseudowire to all the other PE routers. FEC 129 is used for the signaling. The information carried by FEC 129 includes the VPLS ID, the Target Attachment Individual Identifier (TAII) and the Source Attachment Individual Identifier (SAII).

The LDP advertisement also contains the inner label or VPLS label that is expected for the incoming traffic over the pseudowire. This enables the LDP peer to identify the VPLS instance with which the pseudowire is to be associated and the label value that it is expected to use when sending traffic on that pseudowire.

NLRI and Extended Communities

The following figure depicts Network Layer Reachability Information (NLRI) and extended communities (Ext Comms).

Figure 8: NLRI and Extended Communities

NLRI:

| Length (2 octets) |
| Route Distinguisher (8 octets) |
| L2VPN Router ID (4 octets) |

Ext Comms:

| VPLS-ID (8 octets) |
| Route Target (8 octets) |
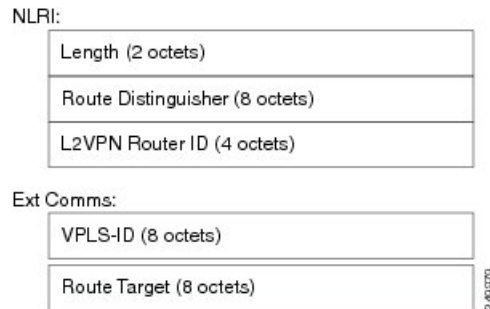
CFM on VPLS

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
CFM on VPLS	Release 7.3.2	This feature helps you monitor and manage a Layer 2 VPN running VPLS. It does so by providing proactive network management, enabling fault detection and isolation, and reporting end-to-end ethernet connectivity issues.

This feature enables you to monitor the VPLS network using connectivity fault management (CFM). CFM is a service-level Operations and Maintenance (OAM) protocol that provides a mechanism for monitoring and troubleshooting end-to-end Ethernet services. This feature provides high speed Layer 2 and Layer 3 services with high resiliency and less operational complexity to different market segments.

CFM on VPLS services support CFM continuity check and ITU-T Y.1731 functions.

The following are the key components of CFM:

- Maintenance Domain (MD)—A maintenance domain is a management space for managing and administering a network. A domain is owned and operated by a single entity and defined by the set of ports internal to the domain and at its boundary. Maintenance Domain divides the network into different operational and administrative domains. For example, customers, service providers, and operators may belong to different domains. For securely maintaining and monitoring an administrative domain, the administrative domain is linked to one maintenance domain. MD supports different domain levels from 0 to 7. A unique maintenance level in the range of 0 to 7 is assigned to each domain by a network administrator. Levels and domain names are useful for defining the hierarchical relationship that exists among domains. The hierarchical relationship of domains parallels the structure of customer, service provider, and operator. The larger the domain, the higher the level value. For example, a customer domain would be larger than an operator domain. The customer domain may have a maintenance level of 7 and the operator domain may have a maintenance level of 0. Typically, operators would have the smallest domains and customers the largest domains, with service provider domains between them in size. All levels of the hierarchy must operate together.

- Maintenance Association/Service (MA)—A maintenance association identifies a service that can be uniquely identified within the maintenance domain. Maintenance association monitors the connectivity of a particular service instance in an MD. MA is defined by a set of Maintenance End Points (MEP) at the edge of a domain.

- Maintenance Point—A maintenance point is a demarcation point on an interface (port) that participates in CFM within a maintenance domain. Maintenance points on device ports act as filters that confine CFM frames within the bounds of a domain by dropping frames that do not belong to the correct level. Maintenance points must be explicitly configured on Cisco devices. Two classes of maintenance points exist, Maintenance End Point (MEP) and Maintenance Intermediate Point (MIP). MIP is not supported.

- Maintenance End Point (MEP)—Is a demarcation point on an interface that participates in CFM within a maintenance domain. MEPs drop all lower-level frames and forward all higher-level frames. MEPs are defined per maintenance domain (level) and service (S-VLAN or ethernet virtual circuit (EVC)). They are at the edge of a domain and define the boundary and confine CFM messages within that boundary. MEPs can proactively transmit CFM continuity check messages (CCMs) and at the request of an administrator, transmit traceroute, and loopback messages.

  The following MEPs are used:

    - Down MEPs are outward-facing MEPs that communicate through the wire. Down MEPs use the port MAC address and not the bridge-domain MAC address. Down MEPs are used for services spanning a single L2 link.

    - Up MEPs are inward-facing MEPs that communicate through the bridge relay function. Up MEP sends and receives CFM frames at its level through the relay function. Up MEPs are commonly used for services across multiple switches for end-to-end connection with xconnect L2 AC interfaces.

The following protocols are used for CFM:

- Continuity Check Protocol—This protocol is used for fault detection and notification and carries the status of port on which MEP is configured. It is unidirectional and requires no response. This protocol is transmitted at a configurable periodic interval by MEP.

- Loopback Protocol—This protocol is used for fault verification. MEP can transmit a unicast loopback message (LBM) to a MEP in the same MA. The receiving MP responds by transforming the LBM into a unicast loopback back reply (LBR) sent back to the originating MEP.

- Linktrace Protocol—This protocol is used for path discovery and fault isolation. MEP can transmit a multicast message (LTM) in order to discover the MPs and path to a MEP in the same MA.

For more information about CFM, see the *Configuring Ethernet OAM* chapter in the *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*.

Restrictions

- Up MEPs and down MEPs are not supported on the same interface. But, multiple MEPs of the same direction are supported. For example, all up MEPs or all down MEPs on the same interface are supported.

- Maintenance Intermediate Point (MIP) is not supported on any interface.

- Hardware offload is not supported.

- MEPs on bundle member interface are limited to maintenance domain level 0.

- Software offload is supported only on bundle member down MEPs.

Supported Offload Types and CCM Timer Values for CFM on VPLS

Depending on where the continuity check messages (CCMs) are processed, offload is categorized into the following types:

- Software offload—When CCMs are processed by the line card CPU, offload type is known as software offload. Software offload is supported only on bundle interface.

- Hardware offload—When CCMs are processed by network processor (NP), offload type is known as hardware offload. Hardware offload is not supported.

- Non-offload—When CCMs are processed by route processor (RP), offload type is known as non-offload.

CCM intervals are the intervals in which CCMs are sent and received. If the CCMs are not received within the configured interval, the CFM MEP goes down. The CCM intervals can be 10m, 1m, 10sec, 1sec, and 100ms.

Note 3.3ms and 10ms offload timers are not supported.

Configure CFM on VPLS

Figure 9: CFM on VPLS: Full Mesh Topology
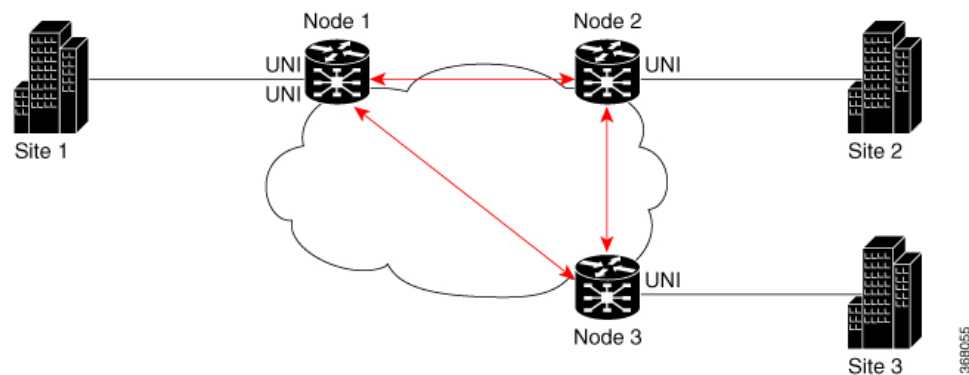
In this topology, node 1, 2 and 3 are PE routers. All nodes are in a VPLS mesh and CFM is configured to monitor the connectivity between them.

Perform the following tasks on PE routers to configure CFM on VPLS:

- Enable CFM service continuity check

- Configure MEP cross-check

- Enable CFM for the interface

Configuration Example

Perform these tasks on the PE routers:

Enable CFM continuity check

```
Router# configure
Router# ethernet cfm
Router(config-cfm# domain vpls_bgp level 3 id null
Router(config-cfm-dmn)# service vpls_bgp_1 bridge group vpls bridge-domain vpls-1 id number
 1000
Router(config-cfm-dmn-svc)# continuity-check interval 10s
```

Repeat the above configurations for node 2 and node 3.

Configure MEP cross-check

```
Router(config-cfm-dmn-svc)# mep crosscheck
Router(config-cfm-dmn-svc-xcheck)# mep crosscheck
Router(config-cfm-dmn-svc-xcheck)# exit
Router(config-cfm-dmn-svc)# log continuity-check errors
Router(config-cfm-dmn-svc)# log continuity-check mep changes
Router(config-cfm-dmn-svc)# commit
```

Repeat the above configurations for node 2 and node 3, with the respective *mep-id* values.

Enable CFM on the interface

```
Router# configure
Router(config)# interface HundredGigE0/0/0/2/0.1000 l2transport
Router(config-subif)# encapsulation dot1q 1000
Router(config-subif)# ethernet cfm
Router(config-if-cfm)# mep domain vpls_bgp service vpls_bgp_1 mep-id 1
Router(config-if-cfm-mep)# commit
```

You must repeat the above configurations for node 2 and node 3, with the respective *mep-id* values.

Running Configuration

This sections shows the running configuration on node 1.

```
ethernet cfm
 domain vpls_bgp level 3 id null
  service vpls_bgp_1 bridge group vpls bridge-domain vpls-1 id number 1000
   continuity-check interval 10s
   mep crosscheck
    mep-id 8191
    !
   log continuity-check errors
   log continuity-check mep changes
  !
!
!
interface HundredGigE0/0/0/2/0.1000 l2transport
 encapsulation dot1q 1000
 ethernet cfm
  mep domain vpls_bgp service vpls_bgp_1 mep-id 1
 !
```

Verification

Verify that you have configured CFM on VPLS successfully. The following output shows the domain ID number along with the maintenance domain level and shows the interface on which the Up MEP is configured.

```
Router(PE1)# show ethernet cfm peer meps
Flags:
```

```
      > - Ok                          I - Wrong interval
      R - Remote Defect received      V - Wrong level
      L - Loop (our MAC received)     T - Timed out
      C - Config (our ID received)    M - Missing (cross-check)
      X - Cross-connect (wrong MAID)  U - Unexpected (cross-check)
      * - Multiple errors received    S - Standby
```

Domain id_no (level 3), Service id_no_vpws_1
Up MEP on TenGigE0/0/0/2/0.1 MEP-ID 1
```
================================================================================
St    ID MAC Address    Port    Up/Downtime   CcmRcvd SeqErr   RDI Error
-- ----- -------------- ------- ----------- ---------- ------ ----- -----
 >  8191 b0c5.3cff.c0c1 Up      00:01:26            9      0     4     0
```

```
Router(PE1)# show ethernet cfm peer meps detail
Domain id_no (level 3), Service id_no_vpws_1
Up MEP on TenGigE0/0/0/2/0.1 MEP-ID 1
================================================================================
```
Peer MEP-ID 8191, MAC b0c5.3cff.c0c1
   CFM state: Ok, for 00:01:44
   Port state: Up
   CCMs received: 11
     Out-of-sequence: **0**
     Remote Defect received: **4**
     Wrong level: **0**
     Cross-connect (wrong MAID): **0**
     Wrong interval: **0**
     Loop (our MAC received):   **0**
     Config (our ID received):   **0**
  Last CCM received 00:00:04 ago:
     Level: 3, Version: 0, Interval: 10s
     Sequence number: 8495, MEP-ID: 8191
     MAID: NULL, UINT: 1
     Chassis ID: Local: Eyrie; Management address: 'Not specified'
     Port status: Up, Interface status: Up

```
Router(PE1)# show ethernet cfm local meps verbose
Domain id_no (level 3), Service id_no_vpws_1
Up MEP on TenGigE0/0/0/2/0.1 MEP-ID 1
================================================================================
  Interface state: Up     MAC address: d46a.355c.b808
  Peer MEPs: 1 up, 0 with errors, 0 timed out (archived)
```
Cross-check errors: 0 missing, 0 unexpected

```
  CCM generation enabled:  Yes, 10s (Remote Defect detected: No)
  AIS generation enabled:  No
  Sending AIS:             No
  Receiving AIS:           No
  Sending CSF:             No
  Receiving CSF:           No

  Packet       Sent     Received
  ------   ----------   ----------------------------------------------------------
  CCM         8508          24  (out of seq: 0)
```

Split-Horizon Groups

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Split-Horizon Groups	Release 7.3.2	This feature prevents packets from going into endless loops by aggregating attachment circuits and pseudowires into one of three groups called split-horizon groups. Split-horizon groups operate on the principle that members within a group will not send traffic to each other thereby preventing traffic loops.

Split horizon is a method for preventing loops in a network by placing forwarding or flooding restrictions between bridge ports based on group membership. The bridge domain aggregates attachment circuits (ACs) and pseudowires (PWs) in one of three groups called split-horizon groups. When applied to bridge domains, split-horizon refers to the flooding and forwarding behavior between members of a split-horizon group. Bridge domain traffic is either unicast or flooding.

Traffic flooding is performed for broadcast, multicast and unknown unicast destination address. Unicast traffic consists of frames sent to bridge-ports where the destination MAC address is known.

Flooding traffic consists of:

- Unknown unicast destination MAC address frames

- frames sent to Ethernet multicast addresses (Spanning Tree BPDUs, and so on)

- Ethernet broadcast frames (MAC address FF-FF-FF-FF-FF-FF)

Members within certain groups are forbidden to send traffic to each other. Members in different groups can send traffic to each other without restriction.

The following table describes how frames received on one member of a split-horizon group are treated and if the traffic is forwarded to the other members of the same split-horizon group. It describes the behavior of forwarding and flooding within and between groups as well as the assignment of Bridge Ports (BPs) to groups:

Table 10: Supported Split-Horizon Groups

Split-Horizon Group	Behavior
0	Default AC group. There is no forwarding and flooding restrictions. Forwards and floods traffic within the group and between all groups. By default, all L2 ACs are added to this group by default. You cannot assign L2 ACs manually through the CLI.

Split-Horizon Group	Behavior
1	Default VFI (core) PW group. Forwarding or flooding of traffic is restricted between bridge ports in this group. Forwarding or flooding of traffic to all other groups is allowed. All VFI PWs are added to this group. You cannot assign VFI PWs manually through the CLI.
2	Optional AC group. Forwarding or flooding of traffic is restricted between bridge ports in this group. Forwarding or flooding traffic to all other groups is allowed. You can manually add ACs, and not VFI PWs using the CLI.

Configuration Example

Perform this task to configure the split-horizon groups feature:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg
Router(config-l2vpn-bg)# bridge-domain bd
Router(config-l2vpn-bg-bd-ac)# interface Ten0/7/0/22/0 <- (split-horizon group 0, default)
Router(config-l2vpn-bg-bd-ac)# interface Ten0/7/0/22/1.1
Router(config-l2vpn-bg-bd-ac)# split-horizon group <- (split-horizon group 2)
Router(config-l2vpn-bg-bd-ac)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-bg-bd-pw)# split-horizon group <- (split-horizon group 2)
Router(config-l2vpn-bg-bd-pw)# vfi vf
Router(config-l2vpn-bg-bd-vfi)# neighbor 172.16.0.1 pw-id 10001 <- (split-horizon group 1,
 default)
Router(config-l2vpn-bg-bd-vfi-pw)# commit
```

Running Configuration

This section shows the split-horizon groups running configuration.

```
l2vpn
  bridge group bg
   bridge-domain bd
    int Ten0/7/0/22/0 <- (split-horizon group 0, default)
    int Ten0/7/0/22/1.1
     split-horizon group <- (split-horizon group 2)
    neighbor 10.0.0.1 pw-id 1
     split-horizon group <- (split-horizon group 2)
    vfi vf
      neighbor 172.16.0.1 pw-id 10001 <- (split-horizon group 1, default)
!
```

Verification

The **show l2vpn bridge-domain detail** command output displays information about bridges, including whether each AC is in the AC split-horizon group or not.

```
Router# show l2vpn bridge-domain detail | i "AC:|Split Horizon|PW:|VFI"
MAC withdraw for Access PW: enabled
```

```
Split Horizon Group: none
P2MP PW: disabled
ACs: 2 (2 up), VFIs: 1, PWs: 2 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
AC: Ten0/7/0/22/0, state is up
Split Horizon Group: none
AC: Ten0/7/0/22/1, state is up
Split Horizon Group: enabled
PW: neighbor 10.0.0.1, pw-id 1, state is up ( established )
Split Horizon Group: enabled
List of VFIs:
VFI vf (up)
PW: neighbor 172.16.0.1, pw-id 10001, state is up ( established )
Split Horizon Group: none
```

Traffic Storm Control

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Traffic Storm Control	Release 7.3.2	This feature monitors incoming traffic levels on a port in the VPLS bridge. It drops traffic when the number of packets reaches the configured threshold level, thus preventing packets from flooding the VPLS bridge and creating excessive traffic and degrading network performance.

When a large number of hosts or routers are attached to the same LAN, there is an increase in protocol traffic rate. The protocol traffic poses a security risk to hosts and routers. When packets flood a VPLS bridge, it creates excessive traffic and degrades network performance. This is commonly known as a traffic storm.

With storm control, you can suppress traffic when the number of packets reaches configured threshold levels, which in turn helps prevent the VPLS bridge disruption and provides Layer 2 port-security under a VPLS bridge. You can configure separate threshold levels for different types of traffic on an access circuit (AC) under a VPLS bridge. Use the **storm-control** command to enable this feature.

How Traffic Storm Control Works?

Storm control monitors incoming traffic levels on a port and drops traffic when the number of packets reaches the configured threshold level during any 1-second interval. The 1-second interval is the monitoring interval and cannot be changed. This 1-second interval is set in the hardware and is not configurable. But, you can configure the number of packets allowed to pass during this interval, per port, and traffic type.

During this interval, the router compares the traffic level with the storm control level that you configured. When the incoming traffic reaches the storm control level configured on the bridge port, storm control drops the traffic until the end of the storm control interval. At the beginning of a new interval, traffic of the specified type is allowed to pass on the port. You can configure the thresholds using a packets-per-second (pps) and kilobit-per-second (kbps) rate.

Feature Behavior

- Storm control configuration is allowed at the bridge port (AC) level.

- Configuration on an L2 subinterface applies to the main interface as well.

- Policer is applied to the main port. Thus, policing applies to all subinterfaces that are within a bridging service.

- When multiple subinterfaces of a common Ethernet port have storm control configured, only one subinterface is used for storm control configuration. Usually, the first subinterface that is added is picked up for storm control configuration.

- Storm control has little impact on router performance. Packets passing through ports are counted regardless of whether the feature is enabled. Additional counting occurs only for the drop counters, which monitor dropped packets. Storm control counts the number of packets dropped per port. The drop counters are cumulative for all traffic types.

- When applied to bundle AC, the policing occurs independently on each main port that is a member of the bundle. Aggregate BUM traffic can be up to the configured rate times the number of bundle members.

Supported Traffic Types for Storm Control

On each VPLS bridge port, both packet per second rate and bits per second rate are supported. You can configure up to three storm control thresholds—one for each of the supported traffic types. If you do not configure a threshold for a traffic type, then storm control is not enabled on that port for that traffic type.

The supported traffic types are:

- Broadcast traffic—Packets with a packet destination MAC address equal to FFFF.FFFF.FFFF.

- Multicast traffic—Packets with a packet destination MAC address not equal to the broadcast address, but with the multicast bit set to 1. The multicast bit is bit 0 of the most significant byte of the MAC address.

- Unknown unicast traffic—Packets with a packet destination MAC address not yet learned.

Restrictions

- Storm control is supported on main ports only.

- Storm control configuration is supported at the bridge-port level, and not at the bridge-domain level.

- PW-level storm control is not supported.

- Storm control is not supported through QoS input policy.

- Although pps is configurable, it is not natively supported. PPS configuration is converted to a kbps value assuming a 256 byte packet size when configuring the hardware policers.

Configure Traffic Storm Control

The storm control feature is disabled by default. It must be explicitly enabled on each port for each traffic type. The thresholds are configured using a packets-per-second (pps) or kilobit-per-second (kbps) rate.

Configuration Example

Perform this task to configure storm control on an attachment circuit (AC).

```
Router# configure
```

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group BG1
Router(config-l2vpn-bg)# bridge-domain BD1
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast kbps 4500
Router(config-l2vpn-bg-bd-ac)# commit
```

Running Configuration

This section shows the storm control running configuration.

```
configure
 l2vpn
  bridge group BG1
   bridge-domain BD1
    interface HundredGigE0/0/0/0
     storm-control broadcast kbps 4500
   !
```

Storm Control Statistics

- The storm control statistics will be present as part of the AC bridging statistics only on the AC where storm control is configured (even if policing occurs from traffic on a different subinterface).

- The storm control statistics will be the aggregate drop statistics across all ACs that share the same main port.

Note   Storm control statistics are available on Line Card based on Q200 Silicon.

Storm control statistics are not available on Line Card based on Q100 Silicon.

GTP Load Balancing

Table 12: Feature History Table

| Feature Name	Release Information	Feature Description
GTP Load Balancing	Release 7.3.2	In addition to the source IP address, destination IP address, and port number, this functionality enables using the unique tunnel endpoint identifier (TEID) to compute load balancing (or hashing) of traffic in tunnels between endpoints. The load balancing occurring at the TEID is unique for each traffic flow and achieves better distribution of traffic over equal-cost links. It also helps in load balancing GTP traffic over bundles at transit routers. By default, this functionality is enabled on the Cisco 8000 Series routers, and you cannot disable it.

What is GTP?

GTP is a tunnel control and management protocol among General Packet Radio Service (GPRS) support nodes. Wireless networks use GTP tunnels to deliver mobile data. GTP includes GTP signaling (GTP-C) and data transfer (GTP-U) procedures. GTP-C specifies a tunnel control and management protocol, and is used to create, delete, and modify tunnels. GTP-U uses a tunneling mechanism to provide a service for carrying user data packets over the network.

What is GTP Load Balancing?

The following figure shows an illustration of the mobile transport GTP-U load balancing.

Figure 10: Mobile Transport GTP-U Load-Balancing
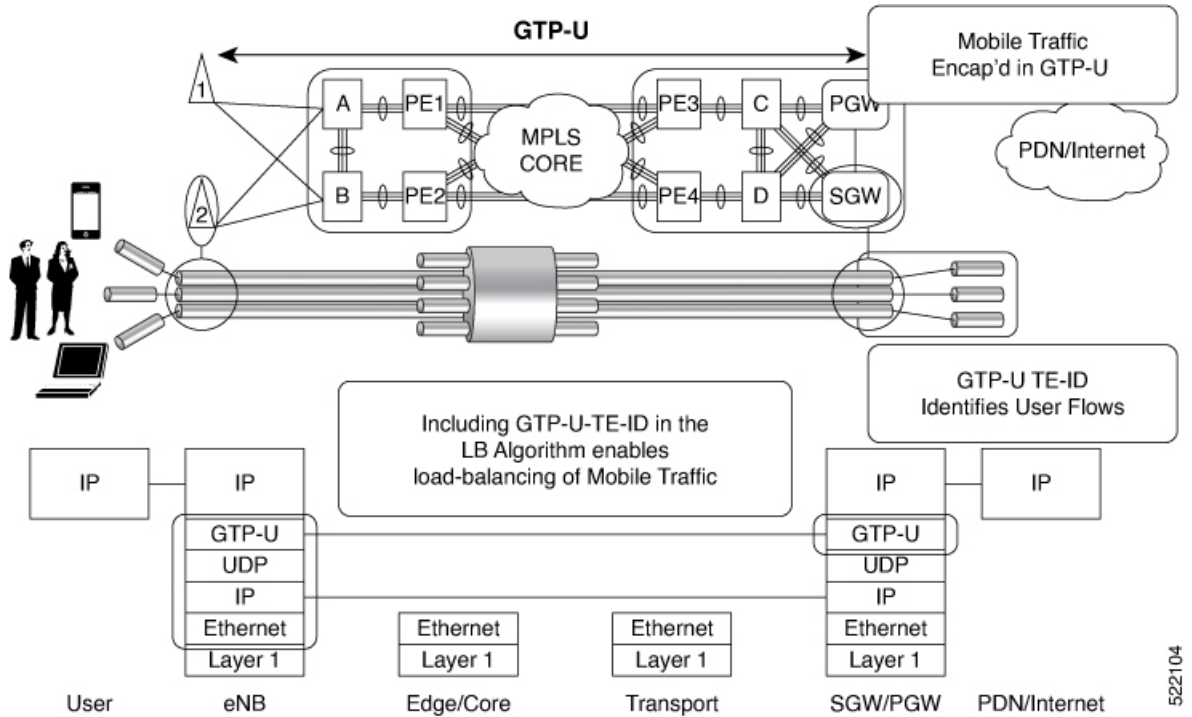
PGW – Packet Data Network Gateway
SGW – Serving Gateway
eNB - evolved Node B

The global L3 flow-based load balancing considers the following fields:

  • Source address

  • Destination address

  • Router ID

  • Source port

  • Destination port

However, for GTP traffic, there are a limited number of unique values for these fields; this causes an uneven distribution of traffic. Sometimes, to facilitate redundancy and load balancing in a network, equal-cost paths exist to different destinations. Load balancing doesn't occur in such scenarios as the source and destination IP addresses and L4 ports are the same.

To achieve a greater distribution of traffic over equal-cost links, the GTP TEID (Tunnel Endpoint ID) in the hash computation algorithm is used. This feature is enabled by default and ensures the load balancing (hashing) computation algorithm includes the GTP TEID, unique for each traffic flow. The GTP load-balancing feature allows efficient distribution of traffic in mobile networks and provides increased reliability and availability for the network.

If the packet is TCP or UDP and the destination port is the GTP-U port (port number 2152), the GTP TEID is considered for loadbalancing.

If TEID is present, load balancing based on tunnel endpoints is supported for Version 1 GTP packet and GTP version 2. For GTP version 0, load balancing occurs only if the fields described earlier have unique values, because there's no TEID in version 0.

Note GTP load balancing is performed only for GTP-U (user data) packets. The GTP-C (control data) packets use a different destination port number of 2123 and hence, are subject to only the global L3 flow-based load balancing.

GTP Load Balancing Guidelines

- GTP load balancing is supported only when the UDP or TCP destination port is 2152.

- GTP load balancing is enabled by default. It cannot be disabled.

- GTP load balancing is performed on IPv4 or IPv6 incoming packets with GTP payloads.

- For MPLS packets with GTP payload, the load balancing hash is based on the label stack and the GTP TEID. The maximum limit on the label stack is 14.

- For IPv4 packets with GTP payload, the load balancing hash is based on Router ID, Source IP, Destination IP, L4 Protocol field, Source Port, Destination Port and GTP TEID.

- For IPv6 packets with GTP payload, the load balance hash is based on Router ID, Source IP, Destination IP, Flow label, L4 Protocol field, Source Port, Destination Port and GTP TEID.

- For GTP hashing, the Cisco 8000 Series routers are transit routers but not GPRS tunnel originators..

Configure Point-to-Point Layer 2 Services

Point-to-point service basically emulates a transport circuit between two end nodes so the end nodes appear to be directly connected over a point-to-point link. This can be used to connect two sites.

This section introduces you to point-to-point Layer 2 services, and also describes the configuration procedures to implement it.

The following point-to-point services are supported:

- Local Switching—A point-to-point internal circuit on a router, also known as local connect. Local switching allows switching of Layer 2 data between two attachment circuits on the same device.

- Attachment circuit—An attachment circuit (AC) is a physical or logical port or circuit that connects a CE device to a PE device.

- Pseudowires—A virtual point-to-point circuit from one PE router to another. Pseudowires are implemented over the MPLS network.

✎

Note   Point-to-point Layer 2 services are also called as MPLS Layer 2 VPNs.

Pseudowire over MPLS

Table 13: Feature History Table

Feature Name	Release Information	Feature Description

Pseudowire over MPLS	Release 7.3.15	This feature allows you to tunnel two L2VPN Provider Edge (PE) devices to transport L2VPN traffic over an MPLS core network. MPLS labels are used to transport data over the pseudowire.

A pseudowire (PW) is a point-to-point connection between two provider edge (PE) devices which connects two attachment circuits (ACs). The two ACs connected at each PE are linked by a PW over the MPLS network, which is the MPLS PW.

PWs provide a common intermediate format to transport multiple types of network services over a Packet Switched Network (PSN) – a network that forwards packets – IPv4, IPv6, MPLS, Ethernet.

Pseudowire over MPLS or Ethernet-over-MPLS (EoMPLS) provides a tunneling mechanism for Ethernet traffic through an MPLS-enabled Layer 3 core network. PW over MPLS encapsulates Ethernet protocol data units (PDUs) using MPLS labels to forward them across the MPLS network.

MPLS traffic disposition load-balancing is based on the source and destination MAC addresses, and source and destination IP addresses. For Release 7.3.3, this criteria is also used for load-balancing PW-to-AC inner MPLS traffic payload (**Ethernet**|**MPLS**|**IP** format). The fix is delivered through CSCwh80721 Software Maintenance Upgrade (SMU).

Load-balancing of L2VPN traffic payload (**Ethernet**|**MPLS**|**Ethernet**|**IP** format) is still based on source and destination MAC addresses.

Topology

Here is an example that showcases how the L2VPN traffic is transported using the PW over MPLS network.
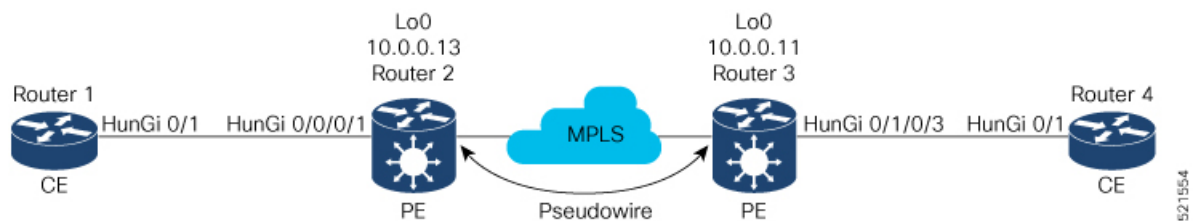
- CEs are connected to PEs using the attachment circuit (AC).

- PW is configured on the PE devices to connect two PEs over an MPLS core network.

Consider a traffic flow from Router 1 to Router 4. Router 1 sends the traffic to Router 2 through the AC. Router 2 adds the MPLS PW label and sends it to Router 3 through the PW. Each PE needs to have an MPLS label in order to reach the loopback of the remote PE. This label, usually called the Interior Gateway Protocol (IGP) label, can be learned through the MPLS Label Distribution Protocol (LDP) or MPLS Traffic Engineering (TE).

One PE advertises the MPLS label to the other PE for PW identification. Router 3 identifies traffic with MPLS label and sends it to the AC connected to Router 4 after removing the MPLS label.

You can configure static or dynamic point-to-point connections.

Configure Static Point-to-Point Connections Using Cross-Connect Circuits

This section describes how you can configure static point-to-point cross connects in a Layer 2 VPN.

Requirements and Limitations

Before you can configure a cross-connect circuit in a Layer 2 VPN, ensure that the following requirements are met:

- The CE and PE routers are configured to operate in a network.

- The name of a cross-connect circuit is configured to identify a pair of PE routers and must be unique within the cross-connect group.

- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect circuit.

- A static virtual circuit local label is globally unique and can be used in only one pseudowire.

Note Static pseudowire connections do not use LDP for signaling.

Topology

The following topology is used to configure static cross-connect circuits in a Layer 2 VPN.

Figure 11: Static Cross-Connect Circuits in a Layer 2 VPN
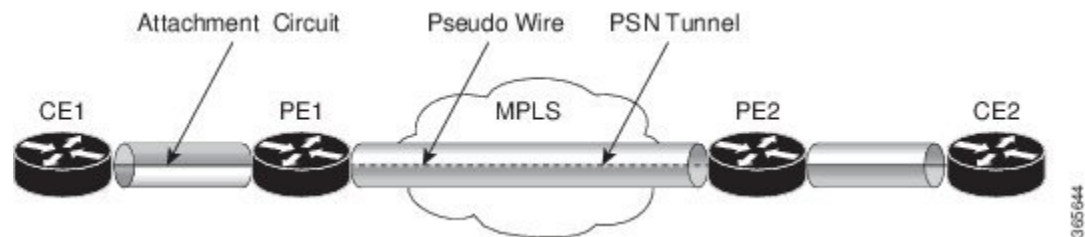
Configuration

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigEt0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.3 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 50 remote 40
Router(config-l2vpn-xc-p2p-pw)# commit

/*Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/2/0/0.4
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.4 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 40 remote 50
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* On PE1 */
!
l2vpn
 xconnect group XCON1
  p2p xc1
   interface HundredGigE0/1/0/0.1
   neighbor ipv4 10.0.0.3 pw-id 100
    mpls static label local 50 remote 40
!

/* On PE2 */
!
l2vpn
 xconnect group XCON2
  p2p xc1
   interface HundredGigE0/2/0/0.4
   neighbor ipv4 10.0.0.4 pw-id 100
    mpls static label local 40 remote 50
  !
```

Verification

```
/* Verify the static cross connect on PE1 */
Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                  Segment 1                        Segment 2
Group       Name     ST   Description         ST           Description          ST
----------------------    ----------------------------     ----------------------------
XCON1       xc1      UP   Hu0/1/0/0.1         UP           10.0.0.3  100    UP
-------------------------------------------------------------------------------------------

/* Verify the static cross connect on PE2 */

Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                  Segment 1                        Segment 2
Group       Name     ST   Description         ST           Description          ST
----------------------    ----------------------------     ----------------------------
XCON2       xc1      UP   Hu0/2/0/0.4         UP           10.0.0.4  100    UP
-------------------------------------------------------------------------------------------
```

Configure Dynamic Point-to-point Cross-Connects

Perform this task to configure dynamic point-to-point cross-connects.

Note For dynamic cross-connects, LDP must be up and running.

Configuration

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group vlan_grp_1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HunGigE 0/0/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
configure
  l2vpn
   xconnect group vlan_grp_1
    p2p vlan1
    interface HunGigE 0/0/0/0.1
    neighbor 10.0.0.1 pw-id 1
!
```

PW over MPLS Supported Modes

The PW over MPLS support these modes:

Ethernet Port Mode

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire VC Type 5	Release 7.3.15	With this feature, Ethernet port mode is supported for pseudowire over MPLS. The virtual connection (VC) type 5 is known as an Ethernet port-based PW. In this mode, both ends of a pseudowire are connected to Ethernet ports and allow a complete ethernet trunk to be transported. The ingress PE transports frames received on a main interface or subinterface. This feature nullifies the need for a dummy tag and reduces overhead. In addition, frame tagging is no longer necessary.

In Ethernet port mode, both ends of a pseudowire are connected to Ethernet ports. In this mode, the port is tunneled over the pseudowire. The ingress PE transports frames received on a main interface or after the subinterface tags are removed when the packet is received on a subinterface. The VLAN manipulation is transported over the type 5 PW, whether tagged or untagged.

This figure shows a sample ethernet port mode packet flow:

Figure 12: Ethernet Port Mode Packet Flow
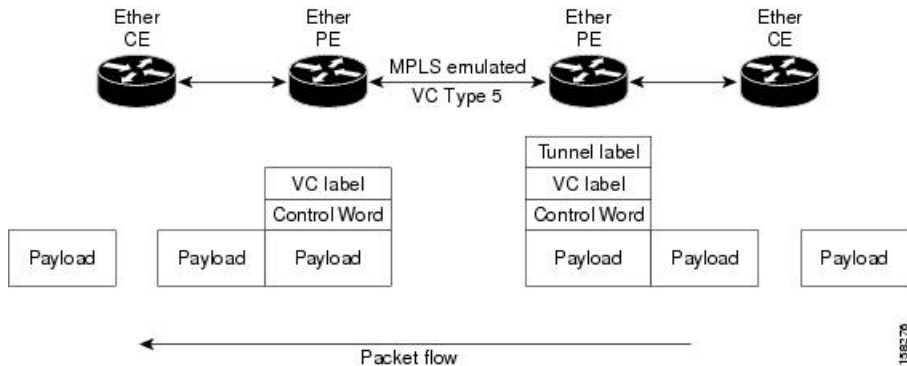
Configure Ethernet Port Mode

Perform this task to configure the Ethernet port mode.

```
/* PE1 configuration */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.11 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# commit

/* PE2 configuration */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/3.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.13 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the Ethernet port mode running configuration.

```
/* PE1 configuration */
l2vpn
 xconnect group grp1
  p2p xc1
   interface HundredGigE0/0/0/1.2
   neighbor 10.0.0.11 pw-id 222

/* PE2 configuration */
l2vpn
 xconnect group grp1
  p2p xc1
   interface HundredGigE0/1/0/3.2
   neighbor 10.0.0.13 pw-id 222
```

Verification

Verify the Ethernet port mode configuration.

The PW type Ethernet indicates a VC type 5 PW.

```
Router# show l2vpn xconnect group grp1 detail
Group grp1, XC xc1, state is up; Interworking none
  AC: HundredGigE0/0/0/1.2, state is up
    Type VLAN; Num Ranges: 1
    VLAN ranges: [2, 2]
    MTU 1504; XC ID 0x840006; interworking none
    Statistics:
      packets: received 186, sent 38448
      bytes: received 12644, sent 2614356
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 10.0.0.11, PW ID 222, state is up ( established )
    PW class not set, XC ID 0xc0000004
    Encapsulation MPLS, protocol LDP
    Source address 10.0.0.13
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set

    PW Status TLV in use
      MPLS          Local                          Remote
      ------------  ------------------------------  -----------------------------
      Label         16026                          16031
      Group ID      0x4000280                      0x6000180
      Interface     HundredGigE0/0/0/1.2           HundredGigE0/1/0/3.2
      MTU           1504                           1504
      Control word  disabled                       disabled
      PW type       Ethernet                       Ethernet
      VCCV CV type  0x2                            0x2
                    (LSP ping verification)        (LSP ping verification)
      VCCV CC type  0x6                            0x6
                    (router alert label)           (router alert label)
                    (TTL expiry)                   (TTL expiry)
      ------------  ------------------------------  -----------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 3221225476
    Create time: 30/03/2021 16:30:58 (21:31:00 ago)
    Last time status changed: 30/03/2021 16:36:42 (21:25:16 ago)
    Statistics:
      packets: received 38448, sent 186
      bytes: received 2614356, sent 12644
```

VLAN Mode

Table 15: Feature History Table

Feature Name	Release Information	Feature Description

Pseudowire VC Type 4	Release 7.3.15	With this feature, VLAN mode is supported for pseudowire over MPLS. A virtual connection (VC) type 4 is the VLAN-based PW. The ingress PE does not remove the incoming VLAN tags that are to be transported over the PW. VC type 4 inserts an extra dummy tag with VLAN 0 onto the frame which is removed on the other side. This mode helps the service provider to segregate traffic for each customer based on the VLAN.

In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4. In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4. VLAN-based (VC Type 4) pseudowires ensure a VLAN tag is transported over the pseudowire by pushing a dummy tag at the attachment circuit ingress. If the rewrite rule pushes two or more tags, a dummy tag is not needed because these VLAN tags are transported over the pseudowire. On the remote router, the dummy tag, if added, is removed before egress.

As illustrated in the following figure, the Ethernet PE associates an internal VLAN tag to the Ethernet port for switching the traffic internally from the ingress port to the pseudowire; however, before moving traffic into the pseudowire, it removes the internal VLAN tag.

Figure 13: VLAN Mode Packet Flow
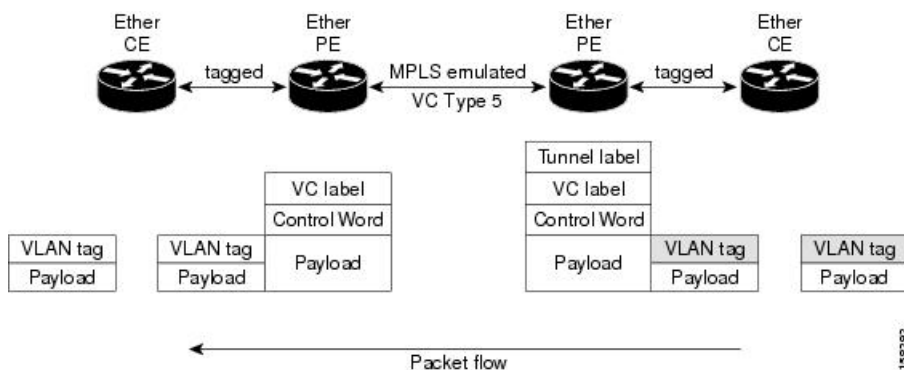
At the egress VLAN PE, the PE associates a VLAN tag to the frames coming out of the pseudowire, and after switching the traffic internally, it sends out the traffic on an Ethernet trunk port.

Note Because the port is in trunk mode, the VLAN PE doesn't remove the VLAN tag and forwards the frames through the port with the added tag.

Limitation

On PW imposition PE, the pushed dummy VLAN Tag Tag Protocol Identifier (TPID) is copied from the TPID of the innermost VLAN tag popped on the ingress L2 interface where traffic is received from. If there is no VLAN tag popped on the L2 interface, the TPID on the dummy VLAN is 0x8100.

On the disposition PE, if the egress VLAN tag push is configured on the egress L2 interface, the innermost pushed VLAN tag TPID is copied from the TPID of the dummy VLAN tag. If there is no egress VLAN push configured on the egress L2 interface, the dummy VLAN tag is discarded.

Configure VLAN Mode

Perform this task to configure VLAN mode.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class VLAN
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode vlan
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.11 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# pw-class VLAN
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the VLAN mode running configuration.

```
l2vpn
 pw-class VLAN
  encapsulation mpls
   transport-mode vlan
  !
 !
 xconnect group grp1
  p2p xc1
   neighbor 10.0.0.11 pw-id 222
    pw-class VLAN
   !
  !
 !
!
```

Verification

Verify the VLAN mode configuration.

The PW type Ethernet VLAN indicates a type 4 PW.

```
Router# show l2vpn xconnect group grp1 detail | i " PW type"
PW type Ethernet VLAN, control word disabled, interworking none
      PW type       Ethernet VLAN                  Ethernet VLAN
```

VLAN Passthrough Mode

Configure the **transport mode vlan passthrough** command under the pw-class to negotiate a virtual connection (VC)-type 4 (Ethernet VLAN) PW, which transports whatever comes out of the AC after the VLAN tag manipulation specified by the **rewrite** command.The VLAN tag manipulation on the EFP ensures that there is at least one VLAN tag left on the frame because you need a VLAN tag on the frame if there are VC-type 4 PWs. No dummy tag 0 is added to the frame when you use the **transport mode vlan passthrough** command.

Pseudowire Redundancy

The Pseudowire Redundancy feature allows you to configure a redundant pseudowire that backs up the primary pseudowire. When the primary pseudowire fails, the PE router switches to the redundant pseudowire. You can elect to have the primary pseudowire resume operation after it becomes functional. The primary pseudowire fails when the PE router fails or when there is a network outage.

Figure 14: Pseudowire Redundancy
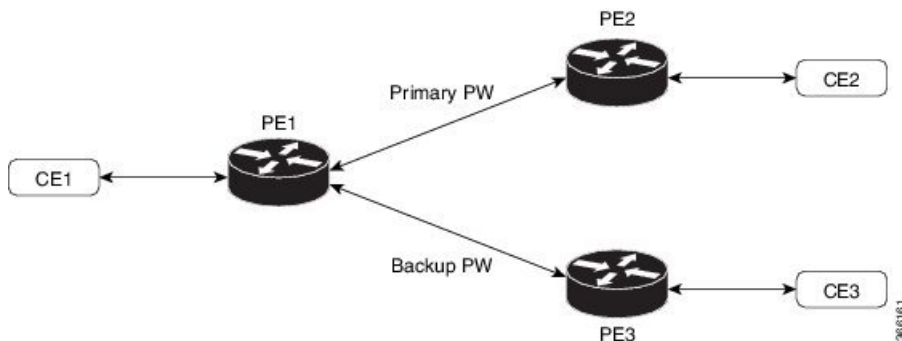
Forcing a Manual Switchover to the Backup Pseudowire

To force the router to switch over to the backup or switch back to the primary pseudowire, use the **l2vpn switchover** command in EXEC mode.

A manual switchover is made only if the peer specified in the command is actually available and the cross-connect moves to the fully active state when the command is entered.

Configure Pseudowire Redundancy

This section describes how you can configure pseudowire redundancy.

You must consider the following restrictions while configuring the Pseudowire Redundancy feature:

  • 2000 active and 2000 backup PWs are supported.

  • Only MPLS LDP is supported.

```
/* Configure PW on PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 192.168.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw-backup)# commit

/* Configure PW on PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

```
/* Configure PW on PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* On PE1 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface HundredGigE 0/1/0/0.1
   neighbor ipv4 172.16.0.1 pw-id 1
    backup neighbor 192.168.0.1 pw-id 1
!

/* On PE2 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface HundredGigE 0/1/0/0.1
   neighbor ipv4 10.0.0.1 pw-id 1

!

/* On PE3 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface HundredGigE 0/1/0/0.1
   neighbor ipv4 10.0.0.1 pw-id 1

!
```

Verification

Verify that the configured pseudowire redundancy is up.

```
/* On PE1 */

Router#show l2vpn xconnect group XCON_1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                    Segment 1                     Segment 2
Group       Name      ST    Description          ST       Description                ST
----------------------  ----------------------------  ----------------------------
XCON_1    XCON1_P2P2 UP    Hu0/1/0/0.1          UP       172.16.0.1       1000    UP
                                                         Backup
                                                         192.168.0.1      1000    SB
--------------------------------------------------------------------------------

/* On PE2 */

Router#show l2vpn xconnect group XCON_1
```

```
Tue Jan 17 15:36:12.327 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                        Segment 1                   Segment 2
Group       Name       ST       Description        ST       Description          ST
----------------------- --------------------------- -----------------------------
XCON_1      XCON1_P2P2 UP       BE100.1            UP       10.0.0.1    1000   UP
------------------------------------------------------------------------------------


/* On PE3 */

Router#show l2vpn xconnect group XCON_1
Tue Jan 17 15:38:04.785 UTC
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                        Segment 1                   Segment 2
Group       Name       ST       Description        ST       Description          ST
----------------------- --------------------------- -----------------------------
XCON_1      XCON1_P2P2 DN       BE100.1            UP       10.0.0.1    1000   SB
------------------------------------------------------------------------------------

Router#show l2vpn xconnect summary
Number of groups: 3950
Number of xconnects: 3950
  Up: 3950  Down: 0  Unresolved: 0 Partially-programmed: 0
  AC-PW: 3950  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
  Up 0 Down 0
  Advertised: 0 Non-Advertised: 0
Number of CE Connections: 0
  Advertised: 0 Non-Advertised: 0
Backup PW:
  Configured   : 3950
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 3950
  Standby Ready: 0
Backup Interface:
  Configured   : 0
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 0
```

Inter-AS Mode

Table 16: Feature History Table

Feature Name	Release Information	Feature Description

| Inter-AS Mode for L2VPN Pseudowire	Release 7.3.15	Inter-AS is a peer-to-peer type that allows VPNs to operate through multiple providers or multi-domain networks using L2VPN cross-connect. This mode allows VPLS autodiscovery to operate across multiple BGP autonomous systems and enables service providers to offer end-to-end VPN connectivity over different geographical locations.

An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and uses a single, clearly defined routing protocol.

As VPNs grow, their requirements expand. In some cases, VPNs need to reside on different autonomous systems in different geographic areas. In addition, some VPNs need to extend across multiple service providers (overlapping VPNs). Regardless of the complexity and location of the VPNs, the connection between autonomous systems must be seamless.

EoMPLS supports a single AS topology where the pseudowire connecting the PE routers at the two ends of the point-to-point EoMPLS cross-connects resides in the same autonomous system; or multiple AS topologies in which PE routers can reside on two different ASs using iBGP and eBGP peering.

The following figure illustrates MPLS over Inter-AS with a basic double AS topology with iBGP/LDP in each AS.

Figure 15: EoMPLS over Inter-AS: Basic Double AS Topology
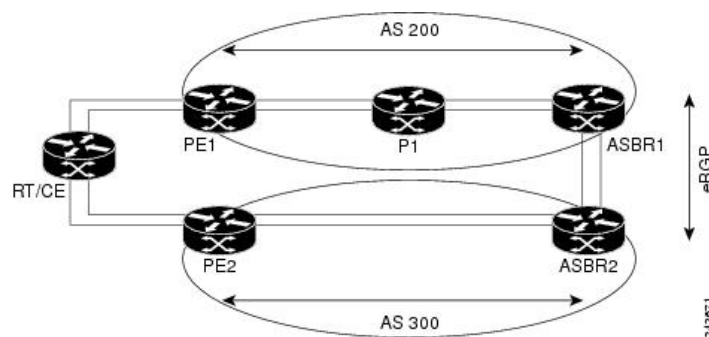
Configure Inter-AS Mode

Perform this task to configure Inter-AS mode:

```
/* PE1 Configuration */
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# router-id 10.0.0.1
Router(config-ldp)# interface HundredGigE0/2/0/3
Router(config-ldp-if)# exit
Router(config-ldp)# router bgp 100
Router(config-bgp)# bgp router-id 10.0.0.1
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
```

```
Router(config-l2vpn)# xconnect group gr1
Router(config-l2vpn-xc)# mp2mp mp1
Router(config-l2vpn-xc-mp2mp)# vpn-id 100
Router(config-l2vpn-xc-mp2mp)# l2-encapsulation vlan
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# rd auto
Router(config-l2vpn-xc-mp2mp-ad)# route-target 2.2.2.2:100
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/1.1 remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit


/* PE2 Configuration */
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# router-id 172.16.0.1
Router(config-ldp)# interface HundredGigE0/3/0/0
Router(config-ldp-if)# exit
Router(config-ldp)# router bgp 100
Router(config-bgp)# bgp router-id 172.16.0.1
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# neighbor 10.0.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group gr1
Router(config-l2vpn-xc)# mp2mp mp1
Router(config-l2vpn-xc-mp2mp)# vpn-id 100
Router(config-l2vpn-xc-mp2mp)# l2-encapsulation vlan
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# rd auto
Router(config-l2vpn-xc-mp2mp-ad)# route-target 2.2.2.2:100
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/2.1 remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/2.2 remote-ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

Running Configuration

This section shows the Inter-AS running configuration.

```
/* PE1 Configuration */
mpls ldp
 router-id 10.0.0.1
 interface HundredGigE0/2/0/3
 !
router bgp 100
 bgp router-id 10.0.0.1
 address-family l2vpn vpls-vpws
 neighbor 172.16.0.1
  remote-as 200
  update-source Loopback0
  address-family l2vpn vpls-vpws
 !
l2vpn
```

```
 xconnect group gr1
  mp2mp mp1
    vpn-id 100
    l2-encapsulation vlan
    autodiscovery bgp
     rd auto
     route-target 2.2.2.2:100
     signaling-protocol bgp
      ce-id 1
        interface HunGigE0/1/0/1.1 remote-ce-id 2
        interface HunGigE0/1/0/1.2 remote-ce-id 3

/* PE2 Configuration */
mpls ldp
 router-id 172.16.0.1
 interface HundredGigE0/3/0/0
 !
router bgp 100
 bgp router-id 172.16.0.1
 address-family l2vpn vpls-vpws
 neighbor 10.0.0.1
  remote-as 100
  update-source Loopback0
  address-family l2vpn vpls-vpws
!
l2vpn
 xconnect group gr1
  mp2mp mp1
    vpn-id 100
    l2-encapsulation vlan
    autodiscovery bgp
     rd auto
     route-target 2.2.2.2:100
     signaling-protocol bgp
      ce-id 2
        interface HunGigE0/1/0/2.1 remote-ce-id 3
        interface HunGigE0/1/0/2.2 remote-ce-id 1
```

Preferred Tunnel Path

Preferred tunnel path functionality lets you map pseudowires to specific traffic-engineering tunnels. Attachment circuits are cross-connected to specific MPLS traffic engineering tunnel interfaces instead of remote PE router IP addresses (reachable using IGP or LDP). Using preferred tunnel path, it is always assumed that the traffic engineering tunnel that transports the L2 traffic runs between the two PE routers (that is, its headend starts at the imposition PE router and its tailend terminates on the disposition PE router).

Note
  • Currently, preferred tunnel path configuration applies only to MPLS encapsulation.

Configure Preferred Tunnel Path

Configuration Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class PATH1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path interface tunnel-te 11 fallback disable
Router(config-l2vpn-pwc-mpls)# commit
```

Configure Local Switching Between Attachment Circuits

Table 17: Feature History Table

Feature Name	Release Information	Feature Description
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 7.3.15	This feature supports tagged or untagged VLAN on physical and bundle interfaces. The tagged VLAN allows you to send and receive the traffic for multiple VLANs whereas, the untagged VLAN allows you to send and receive the traffic for a single VLAN. The multiple VLANs are used to differentiate traffic streams so that the traffic can be split across different services.

Local switching involves the exchange of L2 data from one attachment circuit (AC) to the other. The two ports configured in a local switching connection form an attachment circuit (AC). A local switching connection works like a bridge domain that has only two bridge ports, where traffic enters from one port of the local connection and leaves through the other.

Figure 16: Local Switching Between Attachment Circuits
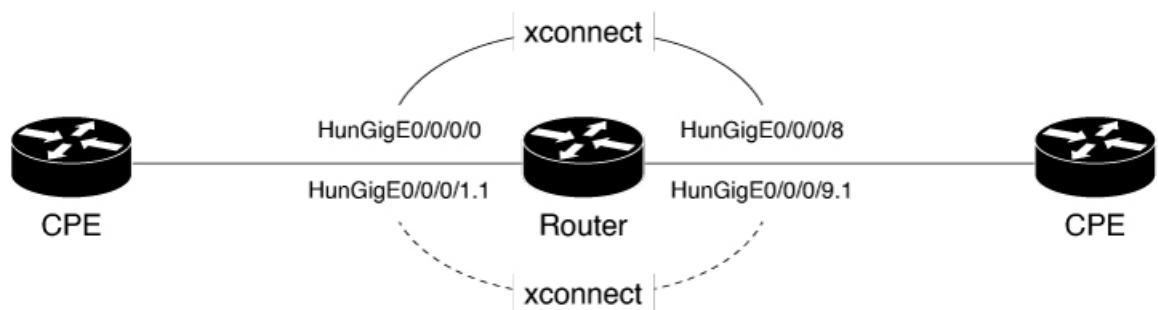
These are some of the characteristics of Layer 2 local switching:

- Because there is no bridging involved in a local connection, there is neither MAC learning nor flooding.

- ACs in a local connection are not in the UP state if the interface state is DOWN.

- Local switching ACs utilize a full variety of Layer 2 interfaces, including Layer 2 trunk (main) interfaces, bundle interfaces, and Ethernet Flow Points (EFPs).

• Same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

Configuration

To configure an AC-AC point-to-point cross connect, complete the following configuration:

• Create Layer 2 interfaces.

• Create a cross-connect group and point-to-point connection.

• Attach the Layer 2 interfaces to point-to-point connection.

```
/* Configure L2 transport and encapsulation on the VLAN sub-interfaces */
Router# configure
Router(config)# interface HunGigE 0/0/0/1.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# exit
Router(config)# interface HunGigE 0/0/0/9.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# commit
```

```
/* Configure local switching on the VLAN sub-interfaces */
Router(config)# l2vpn
Router(config-l2vpn-xc)# p2p XCON1_P2P1
Router(config-l2vpn-xc-p2p)# interface HunGigE0/0/0/1.1
Router(config-l2vpn-xc-p2p)# interface HunGigE0/0/0/9.1
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit
```

Running Configuration

```
configure


interface HunGigE 0/0/0/1.1 l2transport
 encapsulation dot1q 5
!
interface HunGigE 0/0/0/9.1 l2transport
  encapsulation dot1q 5
 !


l2vpn
  p2p XCON1_P2P1
   interface HunGigE0/0/0/1.1
   interface HunGigE0/0/0/9.1
   !
  !
 !
```

Verification

- Verify if the configured cross-connect is UP

```
router# show l2vpn xconnect brief

Locally Switching

   Like-to-Like                            UP        DOWN       UNR

      EFP                                   1          0          0

      Total                                 1          0          0


      Total                                 1          0          0

Total: 1 UP, 0 DOWN, 0 UNRESOLVED
```

MPLS PW Traffic Load Balancing on P Router

When an L2VPN PE needs to send a frame over an MPLS PW, the Ethernet frame is encapsulated into an MPLS frame with one or more MPLS labels; there is at least one PW label and perhaps an IGP label to reach the remote PE.

The MPLS frame is transported by the MPLS network to the remote L2VPN PE. There are typically multiple paths to reach the destination PE:
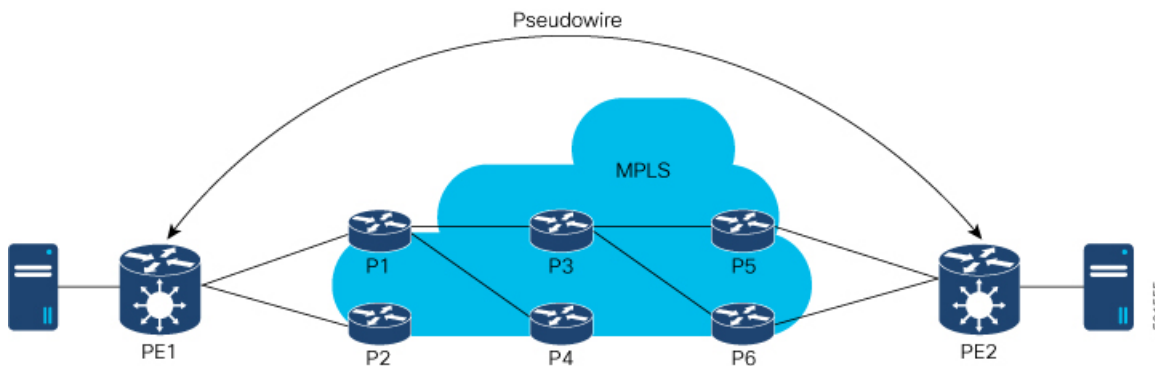
PE1 can choose between P1 and P2 as the first MPLS P router towards PE2. If P1 is selected, P1 then chooses between P3 and P4, and so on. The available paths are based on the IGP topology and the MPLS TE tunnel path.

MPLS service providers prefer to have all links equally utilized rather than one congested link with other underutilized links. This goal is not always easy to achieve because some PWs carry much more traffic than others and because the path taken by a PW traffic depends upon the hashing algorithm used in the core. Multiple high bandwidth PWs might be hashed to the same links, which creates congestion.

A very important requirement is that all packets from one flow must follow the same path. Otherwise, this leads to out-of-order frames, which might impact the quality or the performance of the applications.

Use the following methods to load balance the MPLS PW traffic:

Load Balance MPLS PW Traffic using Control-Word and Flow-Label

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
Load Balance MPLS PW Traffic using Control-Word	Release 7.3.15	This feature allows the router to correctly identify the Ethernet PW packet over an IP packet, thus preventing the selection of wrong equal-cost multipath (ECMP) path for the packet that leads to the misordering of packets. This feature inserts the control word keyword immediately after the MPLS label to separate the payload from the MPLS label over a PW. The control word carries layer 2 control bits and enables sequencing. The **control-word** keyword is added.
Load Balance MPLS PW Traffic using Flow-Label	Release 7.3.15	The flow-label provides the capability to identify individual flows within a pseudowire and provides routers the ability to use these flows to load balance traffic. Individual flows are determined by the hashing algorithm configured under L2VPN. Similar packets with the same source and destination addresses are all said to be in the same flow. A flow-label is created based on indivisible packet flows entering a pseudowire and is inserted as the lowermost label in the packet. Routers can use the flow-label for load balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core. The **flow-label** keyword is added.

Load balancing using Control-Word

If the MPLS packet contains the MAC address that starts with 0x4 or 0x6, a label switching router (LSR) misidentifies the Ethernet PW packet as an IP packet. The router considers that there is an IPv4 or IPv6 packet inside the MPLS packet and tries to load balance based on a hash of the source and destination IPv4 or IPv6 addresses extracted from the frame. This leads to the selection of the wrong equal-cost multipath (ECMP) path for the packet, leading to the misordering of packets.

This must not apply to an Ethernet frame that is encapsulated and transported over a PW because the destination MAC address considers the bottom label.

To overcome this issue, use the **control-word** keyword under a pw-class that is attached to a point-to-point PW. The control word is inserted immediately after the MPLS labels. Pseudowire over MPLS also, known as Ethernet over MPLS (EoMPLS), allows you to tunnel two L2VPN Provider Edge (PE) devices to transport L2VPN traffic over an MPLS cloud. This feature uses MPLS labels to transport data over the PW. The two L2VPN PEs are typically connected at two different sites with an MPLS core between them. This feature allows you to migrate legacy ATM and Frame Relay services to MPLS or IP core without interrupting the existing services.
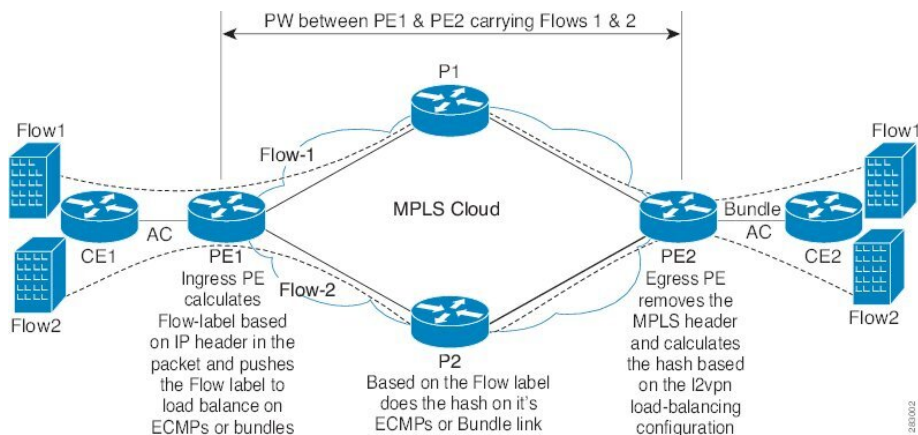
Load balancing using Flow-Label

Routers typically load balance traffic based on the lowermost label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric load balancing. The flow, in this context,

refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) to a destination PE.

The flow-label provides the capability to identify individual flows within a pseudowire and provides routers the ability to use these flows to load balance traffic. A flow-label is created based on individual packet flows entering a pseudowire and is inserted as the lowermost label in the packet. Routers can use the flow-label for load balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.

Topology

This example illustrates two flows distributing over ECMPs and bundle links.

Configure Load balancing using Control-Word and Flow-Label

Perform this task to configure load balancing using the **control-word** and **flow-label**.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class path1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc)# control-word
Router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.2 pw-id 2000
Router(config-l2vpn-xc-p2p-pw)# pw-class path1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the running configuration.

```
l2vpn
 pw-class path1
  encapsulation mpls
   control-word
   load-balancing
```

```
   flow-label both
 !
!
xconnect group grp1
 p2p vlan1
  interface HundredGigE0/0/0/1.2
  neighbor ipv4 10.0.0.2 pw-id 2000
   pw-class path1
   !
```

L2VPN Traffic Load Balancing on PE Router

For AC to AC local switching traffic, load balance is based on flow information including source and destination MAC address, VLAN ID, source and destination IP address, and layer 4 source and destination port ID.

For AC to PW imposition traffic, load balance is based on flow information including source and destination MAC address, VLAN ID, source and destination IP address, and layer 4 source and destination port ID.

The load balancing for PW to AC disposition traffic uses both the source and destination MAC address, as well as the 3/5 tuple information found in the IP Header, which includes the source and destination IP address.

CHAPTER **8**

Integrated Routing and Bridging

This chapter describes the configuration of Integrated Routing and Bridging (IRB). IRB provides the ability to exchange traffic between bridging services and a routed interface using a Bridge-Group Virtual Interface (BVI).

Understanding IRB

IRB provides Layer 2 bridging service between hosts that are within a Layer 2 domain. Also, it provides routing service for hosts that are in different subnets within a Layer 3 VPN.

Bridge-Group Virtual Interface

The BVI is a virtual interface within the router that acts like a normal routed interface. A BVI is associated with a single bridge domain and represents the link between the bridging and the routing domains on the router. To support receipt of packets from a bridged interface that are destined to a routed interface, the BVI must be configured with the appropriate IP addresses and relevant Layer 3 attributes. The BVI does not support bridging itself, but acts as a gateway for the corresponding bridge-domain to a routed interface within the router.

BVI supports these attributes, and has the following characteristics:

- Uses a MAC address taken from the local chassis MAC address pool, unless overridden at the BVI interface.

- Is configured as an interface type using the **interface bvi** command and uses an IPv4 or IPv6 address that is in the same subnet as the hosts on the segments of the bridged domain.

- The BVI identifier is independent of the bridge-domain identifier.

- BVI interfaces support a number range of 1 to 4294967295.

BVI Interface and Line Protocol States

Like typical interface states on the router, a BVI has both an Interface and Line Protocol state.

The BVI interface state is Up when the following occurs:

- The BVI interface is created.

- The bridge-domain that is configured with the **routed interface bvi** command has at least one available active bridge port (Attachment circuit [AC]).

Note   A BVI will be moved to the Down state if all of the bridge ports (Ethernet flow points [EFPs]) associated with the bridge domain for that BVI are down. However, the BVI will remain up if at least one bridgeport is up, even if all EFPs are down.

These characteristics determine when the the BVI line protocol state is up:

- The bridge-domain is in Up state.

- The BVI IP address is not in conflict with any other IP address on another active interface in the router.

Prerequisites for Configuring IRB

Before configuring IRB, ensure that these tasks and conditions are met:

- Know the IP addressing and other Layer 3 information to be configured on the bridge virtual interface (BVI).

- Complete MAC address planning if you decide to override the common global MAC address for all BVIs.

  You can replace the preferred MAC address for the BVI interface with the default MAC address allocated from the chassis pool. The MAC address is divided into:

  - 32 bits most significant bits called MAC prefix.

    The router has a limitation of four different MAC prefixes per system. You must not use more than four different MAC prefixes when choosing the MAC address for BVI and other L3 interfaces.

  - 16 bits least significant called MAC host. You can choose any value for the MAC host.

- Be sure that the BVI network address is being advertised by running static or dynamic routing on the BVI interface.

Packet Flows Using IRB

This figure shows a simplified functional diagram of an IRB implementation to describe different packet flows between Host A, B, and C. In this example, Host C is on a network with a connection to the same router. In reality, another router could be between Host C and the router shown.

Figure 17:

Figure 18: IRB Packet Flows Between Hosts
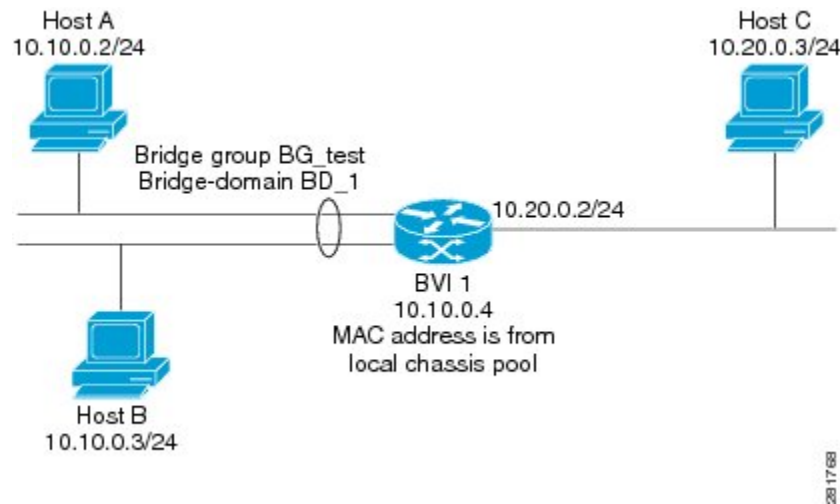
When IRB is configured on a router, the following processing happens:

- ARP requests are resolved between the hosts and BVI that are part of the bridge domain.

- All packets from a host on a bridged interface go to the BVI if the destination MAC address matches the BVI MAC address. Otherwise, the packets are bridged.

- For packets destined for a host on a routed network, the BVI forwards the packets to the routing engine before sending them out a routed interface.

- For packets that are destined for a host on a segment in the bridge domain that come in to the router on a routed interface, the BVI forwards the packet to the bridging engine, which forwards it through the appropriate bridged interface.

Packet Flows When Host A Sends to Host B on the Bridge Domain

When Host A sends data to Host B in the bridge domain on the 10.10.0.0 network, no routing occurs. The hosts are on the same subnet and the packets are bridged between their segment interfaces on the router.

Packet Flows When Host A Sends to Host C From the Bridge Domain to a Routed Interface

Using host information from this figure, the following occurs when Host A sends data to Host C from the IRB bridging domain to the routing domain:

- Host A sends the packet to the BVI (as long any ARP request the is resolved between the host and the BVI). The packet has the following information:

    - Source MAC address of host A.

    - Destination MAC address of the BVI.

- Since Host C is on another network and needs to be routed, the BVI forwards the packet to the routed interface with the following information:

    - IP source MAC address of Host A (10.10.0.2) is changed to the MAC address of the BVI (10.10.0.4).

- IP destination address is the IP address of Host C (10.20.0.3).

- Interface 10.20.0.2 sees receipt of a packet from the routed BVI 10.10.0.4. The packet is then routed through interface 10.20.0.2 to Host C.

Packet Flows When Host C Sends to Host B From a Routed Interface to the Bridge Domain

Using host information from this figure, the following occurs when Host C sends data to Host B from the IRB routing domain to the bridging domain:

- The packet comes into the routing domain with the following information:

  - MAC source address—MAC of Host C.

  - MAC destination address—MAC of the 10.20.0.2 ingress interface.

  - IP source address—IP address of Host C (10.20.0.3).

  - IP destination address—IP address of Host B (10.10.0.3).

- When interface 10.20.0.2 receives the packet, it looks in the routing table and determines that the packet needs to be forwarded to the BVI at 10.10.0.4.

- The routing engine captures the packet that is destined for the BVI and forwards it to the BVI's corresponding bridge domain. The packet is then bridged through the appropriate interface if the destination MAC address for Host B appears in the bridging table, or is flooded on all interfaces in the bridge group if the address is not in the bridging table.

Configure IRB

Follow these steps to configure an IRB:

- Configure the Bridge Group Virtual Interface

- (Optional) Configure the static MAC address on the BVI interface

- Configure the Layer 2 AC Interfaces

- Configure a Bridge Group and Assigning Interfaces to a Bridge Domain

- Associate the BVI as the Routed Interface on a Bridge Domain

Configuration Example

```
/* Configure the BVI and its IPv4 address * /
Router# configure
Router(config)#interface bvi 1
Router(config-if)#ipv4 address 10.10.0.4 255.255.255.0
Router(config-if)#ipv6 address 2001:100:1:1::1/96

/* optionally, you can configure the static MAC address */
Router(config-if))# mac-address 2001.100.2
Router(config-if))# exit
!
```

```
/* Configure the Layer 2 AC interface */
Router(config)# interface HundredGigE 0/0/0/1 l2transport
Router(config-if-l2))# exit
Router(config-if))# exit
!

/* Configure the L2VPN bridge group and bridge domain and assign interfaces */
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 10
Router(config-l2vpn-bg)# bridge-domain 1
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/1
Router(config-l2vpn-bg-bd-ac)# exit
!
/* Associate a BVI to the bridge domain */
Router(config-l2vpn-bg-bd)# routed interface bvi 1
Router(config-l2vpn-bg-bd-bvi)# commit


/* IRB configuration for tagged bridge ports (sub-interfaces) in a bridge domain with BVI
*/
Router# configure
Router(config)# interface HundredGigE 0/0/0/2.1 l2transport
Router(config-subif)# encapsulation dot1q 102
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
Router(config)# interface bvi 2
Router(config-if)# ipv4 address 56.78.100.1 255.255.255.0
Router(config-if)# ipv6 address 56:78:100::1/64
Router(config-if))# mac-address 2002.100.1
Router(config-if))# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 10
Router(config-l2vpn-bg)# bridge-domain 2
Router(config-l2vpn-bg-bd)# interface HundredGigE 0/0/0/2.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface bvi 2
Router(config-l2vpn-bg-bd-bvi)# commit
```

Note Double VLAN tagged sub-interface is not supported for IRB service.

Verification

Verify the interface status, line protocol state, and packet counters for the specified BVI:

```
Router# show interfaces bvi 1  brief
BVI1 is up, line protocol is up
Interface state transitions: 701
Hardware is Bridge-Group Virtual Interface, address is 2001.0100.0001
Internet address is 10.10.0.4/24
MTU 1514 bytes, BW 10000000 Kbit (Max: 10000000 Kbit)
reliability 255/255, txload 0/255, rxload 1/255
Encapsulation ARPA, loopback not set,
Last link flapped 2d06h
ARP type ARPA, ARP timeout 04:00:00
Last input 00:00:00, output 00:00:13
Last clearing of "show interface" counters 3d18h
30 second input rate 43721000 bits/sec, 49684 packets/sec
30 second output rate 0 bits/sec, 0 packets/sec
15428019162 packets input, 1697081244790 bytes, 0 total input drops
0 drops for unrecognized upper-level protocol
```

```
Received 0 broadcast packets, 0 multicast packets
6084259298 packets output, 669870073726 bytes, 0 total output drops
Output 0 broadcast packets, 0 multicast packets
```

Multiple Spanning Tree Protocol

This chapter introduces you to Multiple Spanning Tree Protocol (MSTP) which is one of the variants of Spanning Tree Protocol (STP) and describes how you can configure the MSTP feature.

Multiple Spanning Tree Protocol

The Multiple Spanning Tree Protocol (MSTP) is a Spanning Tree Protocols (STPs) variant that allows you to create multiple and independent spanning trees over the same physical network. You can configure the parameters for each spanning tree separately. You can select different network devices as the root bridge or different paths to form the loop-free topology. Therefore, you can block a given physical interface for some of the spanning trees and unblock for others.

After setting up multiple spanning tree instances, you can partition the set of VLANs in use. For example, you can assign VLANs 1–100 to spanning tree instance 1, VLANs 101–200 to spanning tree instance 2, VLANs 201–300 to spanning tree instance 3, and so on. Since each spanning tree has a different active topology with different active links, this has the effect of dividing the data traffic among the available redundant links based on the VLAN—a form of load balancing.

MSTP Supported Features

The Cisco 8000 Series Routers support MSTP, as defined in IEEE 802.1Q-2005, on physical Ethernet interfaces and Ethernet Bundle interfaces.This includes the Port Fast and bridge protocol data unit (BPDU) Guard features to break L2 loop. The routers can operate in either standard 802.1Q mode, or in Provide Edge (802.1ad) mode. In provider edge mode, a different MAC address is used for bridge protocol data units (BPDUs), and any BPDUs received with the 802.1Q MAC address are forwarded transparently.

When you have not configured the **allow-bpdu-guard** command on MST default instance, and if one of the bridge ports receives legacy BPDU, the port enters **error-disable** state.

BPDU Guard

The BPDU Guard feature allows you to protect against misconfiguration of edge ports. It is an enhancement to the MSTP port fast feature. When you configure port fast on an interface, MSTP considers that interface to be an edge port and removes it from consideration when calculating the spanning tree. When you configure BPDU Guard, MSTP additionally shuts down the interface using error-disable when an MSTP BPDU is received.

Restrictions

These restrictions apply when using MSTP:

- You can configure MSTP only on the main (L3 or L2 interface) interface.

- The subinterfaces are mapped to MSTI instances by the outermost VLAN tag ID, even if the subinterface encapsulation is a QinQ or a single VLAN tag.

- There's no intersection with split-horizon group alignment and MSTI grouping using VLAN ID. Each grouping runs independently.

- All subinterfaces in a bridge domain must use the same MSTI when MSTP is running on the corresponding main interfaces.

- When MSTP runs on a main interface, untagged subinterface shouldn't be created.

Configure MSTP

By default, STP is disabled on all interfaces. You must enable MSTP on each physical or Ethernet Bundle interface. When you configure MSTP on an interface, all the subinterfaces of that interface are automatically MSTP-enabled.

Perform these tasks to configure MSTP:

- Configure VLAN interfaces

- Configure L2VPN bridge-domains

- Configure MSTP parameters

Configuration Example

```
/* Configure VLAN interfaces */
Router# configure
Router(config)# interface HundredGigE0/0/0/2.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface HundredGigE0/0/0/3.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface HundredGigE0/0/0/14.1001 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config)# interface HundredGigE0/0/0/2.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config)# interface HundredGigE0/0/0/3.1021 l2transport
```

```
Router(config-subif)# encapsulation dot1q 1021
Router(config)# interface HundredGigE0/0/0/14.1021 l2transport
Router(config-subif)# encapsulation dot1q 1021
Router(config-subif)# commit

/* Configure L2VPN bridge-domains */
Router# configure
Router(config)# l2vpn bridge group mstp
Router(config-l2vpn-bg)# bridge-domain mstp1001
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/2.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/3.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/14.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn-bg)# bridge-domain mstp1021
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/2.1021
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/3.1021
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# int HundredGigE 0/0/0/14.1021
Router(config-l2vpn-bg-bd-ac)# commit

/* Configure MSTP Parameters */
Router#configure
Router(config)#spanning-tree mst a
Router(config-mstp)#interface HundredGigE0/0/0/2
Router(config-mstp-if)#portfast bpduguard
Router(config-mstp-if)#commit
```

Running Configuration

This section show MSTP running configuration.

```
!
Configure
/* Configure VLAN interfaces */
interface HundredGigE0/0/0/2.1001 l2transport
 encapsulation dot1q 1001
!
interface HundredGigE0/0/0/3.1001 l2transport
 encapsulation dot1q 1001
!
interface HundredGigE0/0/0/14.1001 l2transport
 encapsulation dot1q 1001

interface HundredGigE0/0/0/2.1021 l2transport
 encapsulation dot1q 1021
!
interface HundredGigE0/0/0/3.1021
 l2transport
 encapsulation dot1q 1021
!
interface HundredGigE0/0/0/14.1021 l2transport
 encapsulation dot1q 1021
!
/* Configure L2VPN Bridge-domains */
l2vpn
 bridge group mstp
  bridge-domain mstp1001
```

```
    interface HundredGigE0/0/0/2.1001
    !
    interface HundredGigE0/0/0/3.1001
    !
    interface HundredGigE0/0/0/14.1001
    !
bridge-domain mstp1021
    interface HundredGigE0/0/0/2.1021
    !
    interface HundredGigE0/0/0/3.1021
    !
    interface HundredGigE0/0/0/14.1021
!
/* Configure MSTP Parameters */
spanning-tree mst a
 interface HundredGigE0/0/0/2
   portfast bpduguard
!
!
```

Information About Multiple Spanning Tree Protocol

To configure Ethernet services access lists, you must understand these concepts:

Spanning Tree Protocol Overview

Ethernet is no longer just a link-layer technology used to interconnect network vehicles and hosts. Its low cost and wide spectrum of bandwidth capabilities coupled with a simple *plug and play* provisioning philosophy have transformed Ethernet into a legitimate technique for building networks, particularly in the access and aggregation regions of service provider networks.

Ethernet networks lacking a TTL field in the Layer 2 (L2) header and, encouraging or requiring multicast traffic network-wide, are susceptible to broadcast storms if loops are introduced. However, loops are a desirable property as they provide redundant paths. Spanning tree protocols (STP) are used to provide a loop free topology within Ethernet networks, allowing redundancy within the network to deal with link failures.

There are many variants of STP; however, they work on the same basic principle. Within a network that may contain loops, a sufficient number of interfaces are disabled by STP so as to ensure that there is a loop-free spanning tree, that is, there is exactly one path between any two devices in the network. If there is a fault in the network that affects one of the active links, the protocol recalculates the spanning tree so as to ensure that all devices continue to be reachable. STP is transparent to end stations which cannot detect whether they are connected to a single LAN segment or to a switched LAN containing multiple segments and using STP to ensure there are no loops.

STP Protocol Operation

All variants of STP operate in a similar fashion: STP frames (known as bridge protocol data units (BPDUs)) are exchanged at regular intervals over Layer 2 LAN segments, between network devices participating in STP. Such network devices do not forward these frames, but use the information to construct a loop free spanning tree.

The spanning tree is constructed by first selecting a device which is the *root* of the spanning tree (known as the root bridge), and then by determining a loop free path from the *root bridge* to every other device in the network. Redundant paths are disabled by setting the appropriate ports into a blocked state, where STP frames can still be exchanged but data traffic is never forwarded. If a network segment fails and a redundant path

exists, the STP protocol recalculates the spanning tree topology and activates the redundant path, by unblocking the appropriate ports.

The selection of the root bridge within a STP network is determined by the lowest Bridge ID which is a combination of configured bridge priority and embedded mac address of each device. The device with the lowest priority, or with equal lowest priority but the lowest MAC address is selected as the root bridge.

Root port: is selected based on lowest root path cost to root bridge. If there is a tie with respect to the root path cost, port on local switch which receives BPDU with lowest sender bridge ID is selected as root port.

Designated port: Least cost port on local switch towards root bridge is selected as designated port. If there is a tie, lowest number port on local switch is selected as designated port.

The selection of the active path among a set of redundant paths is determined primarily by the port path cost. The port path cost represents the cost of transiting between that port and the root bridge - the further the port is from the root bridge, the higher the cost. The cost is incremented for each link in the path, by an amount that is (by default) dependent on the media speed. Where two paths from a given LAN segment have an equal cost, the selection is further determined by the lowest bridge ID of the attached devices, and in the case of two attachments to the same device, by the configured port priority and port ID of the neighboring attached ports.

Once the active paths have been selected, any ports that do not form part of the active topology are moved to the blocking state.

Variants of STP

The following are the supported variants of the Spanning Tree Protocol:

- Legacy STP (STP)—The original STP protocol was defined in IEEE 802.1D-1998. This creates a single spanning tree which is used for all VLANs and most of the convergence is timer-based.

- Multiple STP (MSTP)—A further enhancement was defined in IEEE 802.1Q-2005. This allows multiple spanning tree instances to be created over the same physical topology. By assigning different VLANs to the different spanning tree instances, data traffic can be load-balanced over different physical links. The number of different spanning tree instances that can be created is restricted to a much smaller number than the number of possible VLANs; however, multiple VLANs can be assigned to the same spanning tree instance. The BPDUs used to exchange MSTP information are always sent untagged; the VLAN and spanning tree instance data is encoded inside the BPDU.

MSTP Regions

Along with supporting multiple spanning trees, MSTP also introduces the concept of regions. A region is a group of devices under the same administrative control and have similar configuration. In particular, the configuration for the region name, revision, and the mapping of VLANs to spanning tree instances must be identical on all the network devices in the region. A digest of this information is included in the BPDUs sent by each device, so as to allow other devices to verify whether they are in the same region.

The following figure shows the operation of MST regions when bridges running MSTP are connected to bridges running legacy STP or RSTP. In this example, switches SW1, SW2, SW3, SW4 support MSTP, while switches SW5 and SW6 do not.

Figure 19: MST Interaction with Non-MST Regions
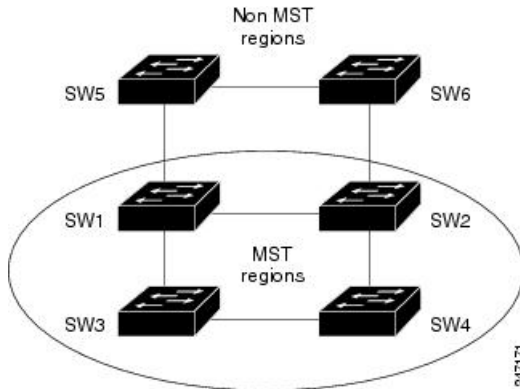
To handle this situation, an Internal Spanning Tree (IST) is used. This is always spanning tree instance 0 (zero). When communicating with non-MSTP-aware devices, the entire MSTP region is represented as a single switch. The logical IST topology in this case is shown in the following figure.

Figure 20: Logical Topology in MST Region Interacting with Non-MST Bridges
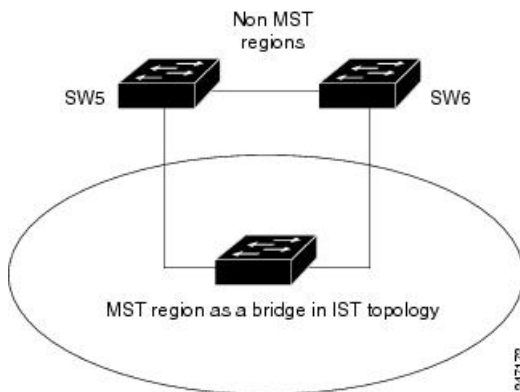
The same mechanism is used when communicating with MSTP devices in a different region. For example, SW5 in the above figure could represent a number of MSTP devices, all in a different region compared to SW1, SW2, SW3 and SW4.

MSTP Port Fast

MSTP includes a *Port Fast* feature for handling ports at the edge of the switched Ethernet network. For devices that only have one link to the switched network (typically host devices), there is no need to run MSTP, as there is only one available path. Furthermore, it is undesirable to trigger topology changes (and resultant MAC flushes) when the single link fails or is restored, as there is no alternative path.

By default, MSTP monitors ports where no BPDUs are received, and after a timeout, places them into *edge mode* whereby they do not participate in MSTP. However, this process can be speeded up (and convergence of the whole network thereby improved) by explicitly configuring edge ports as port fast.

Note
- You must disable and re-enable the port for Port Fast configuration to take effect. Use **shutdown** and **no shutdown** command (in interface configuration mode) to disable and re-enable the port.

- Port Fast is implemented as a Cisco-proprietary extension in Cisco implementations of legacy STP. However, it is encompassed in the standard for MSTP, where it is known as Edge Port.

MSTP Root Guard

In networks with shared administrative control, it may be desirable for the network administrator to enforce aspects of the network topology and in particular, the location of the root bridge. By default, any device can become the root bridge for a spanning tree, if it has a lower priority or bridge ID. However, a more optimal forwarding topology can be achieved by placing the root bridge at a specific location in the centre of the network.

Note
The administrator can set the root bridge priority to 0 in an effort to secure the root bridge position; however, this is no guarantee against another bridge which also has a priority of 0 and has a lower bridge ID.

The root guard feature provides a mechanism that allows the administrator to enforce the location of the root bridge. When root guard is configured on an interface, it prevents that interface from becoming a root port (that is, a port via which the root can be reached). If superior information is received via BPDUs on the interface that would normally cause it to become a root port, it instead becomes a backup or alternate port. In this case, it is placed in the blocking state and no data traffic is forwarded.

The root bridge itself has no root ports. Thus, by configuring root guard on every interface on a device, the administrator forces the device to become the root, and interfaces receiving conflicting information are blocked.

Note
Root Guard is implemented as a Cisco-proprietary extension in Cisco implementations of legacy STP. However, it is encompassed in the standard for MSTP, where it is known as Restricted Role.

MSTP Topology Change Guard

In certain situations, it may be desirable to prevent topology changes originating at or received at a given port from being propagated to the rest of the network. This may be the case, for example, when the network is not under a single administrative control and it is desirable to prevent devices external to the core of the network from causing MAC address flushing in the core. This behavior can be enabled by configuring Topology Change Guard on the port.

Note
Topology Change Guard is known as *Restricted TCN* in the MSTP standard.