L2VPN Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 24.1.x , 24.2.x , 24.3.x

First Published: 2024-03-14

Last Modified: 2024-06-01

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Preface

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

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

Table 1: Changes to This Document

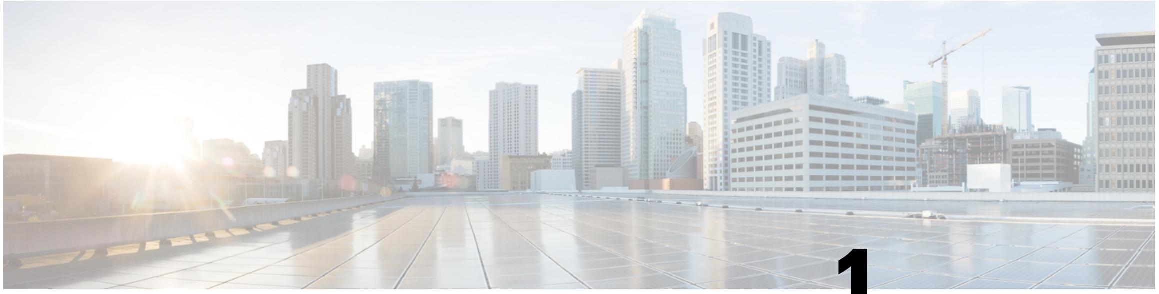
Date	Summary
September 2024	Republished with feature updates for Release 24.3.1.
June 2024	Republished with feature updates for Release 24.2.11.
March 2024	Initial release of this document.

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.



CHAPTER 1

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 24.x.x

Feature	Description	Changed in Release	Where Documented
Pseudowire Headend	This feature was introduced	Release 24.3.1	Pseudowire Headend, on page 83
Traffic Mirroring on PWHE	This feature was introduced	Release 24.3.1	Traffic Mirroring on PWHE, on page 91
Withdraw Dynamic MAC Addresses Between Peer PE Routers	This feature was introduced	Release 24.2.11	Withdraw Dynamic MAC Addresses Between Peer PE Routers, on page 40
G.8032 Ethernet Ring Protection Switching	This feature was introduced	Release 24.2.11	G.8032 Ethernet Ring Protection Switching, on page 121
Dot1Q Q-in-Q (0x8100/0x8100) Tunneling for VLAN Subinterface Encapsulation	This feature was introduced	Release 24.1.1	Introduction to Virtual LANs in Layer 2 VPNs

CHAPTER 2

YANG Data Models for VPN Features

This chapter provides information about the YANG data models for L2VPN and Ethernet Services.

Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the **Available-Content.md** file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.

CHAPTER 3

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.

Line cards and routers with the Q100, Q200, and P100 based Silicon One ASIC support this feature.

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 4

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

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

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

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

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

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

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

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

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

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

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

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

Running Configuration

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

Verification

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

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

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

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

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

Running Configuration

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

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

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

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

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

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

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

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

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

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

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

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 5

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 6

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.

CHAPTER 7

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

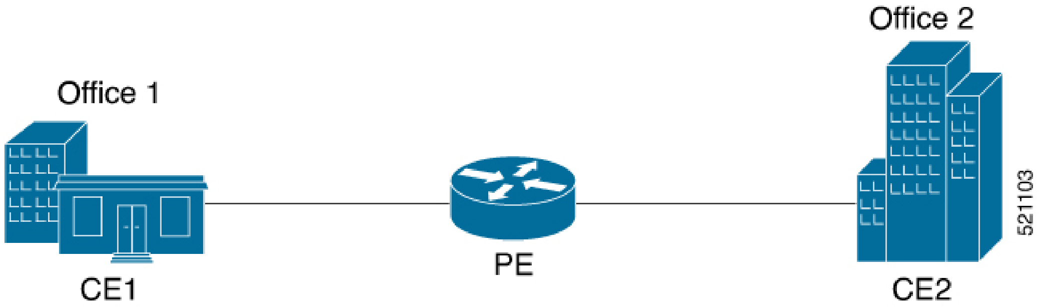
VLAN bridging is the simplest mode of L2 bridging. In this mode, all traffic that is received on the switch is either Ethernet II frames or IEEE 802.3 frames.

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.

Withdraw Dynamic MAC Addresses Between Peer PE Routers

Table 7: Feature History Table

Feature Name	Release Information	Feature Description

Withdraw Dynamic MAC Addresses Between Peer PE Routers	Release 24.2.11	We now prevent packet drops between peer routers when the attachment circuit (AC) of a PE router goes down, by withdrawing all dynamic MAC addresses from that PE router. When the AC goes down, the PE routers remove or unlearn the MAC addresses learned from the peer routers, that do not need to be relearned. This enables faster convergence when the AC comes up.

By withdrawing dynamic MAC addresses, the packet drops between peer routers are prevented when the AC of a PE router goes down. This feature uses Label Distribution Protocol (LDP)-based MAC address withdrawal message. A MAC address list Type Length Value (TLV) is part of the MAC address withdrawal message.

This feature optimizes MAC address withdrawal. The optimization allows PEs to retain the MAC addresses that are learned from the CE devices over the access side. When the AC goes down, only the MAC addresses that are learned from peer PEs are cleared out. As there is no need for the PE to relearn the cleared MAC addresses, faster convergence is achieved when the AC comes up.

The MAC address withdrawal is enabled by default. Use the **mac withdraw disable** command to disable MAC address withdrawal.

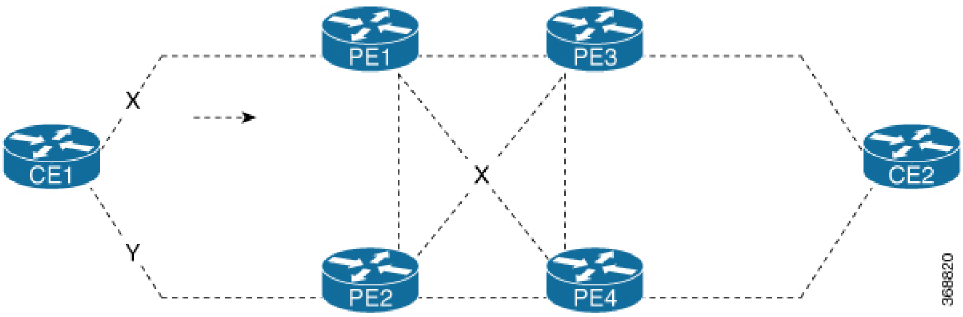
Topology

Consider the following topology in which CE1 is dual-homed to PE1 and PE2. The link X is active and Y is a redundant link. Initially PE1, PE2, PE3, and PE4 learn their MAC address forwarding tables that are based on the traffic profile and traffic becomes a known unicast. By default, the MAC address withdrawal feature is enabled on all the PEs. The PEs clear MAC address entries when they receive MAC address withdrawal message.

The following are the MAC address withdrawal messages that are sent based on the status of link:

- Scenario 1: When link X, which is the AC of PE1 goes down, PE1 sends an LDP MAC withdrawal TLV message “FLUSH ALL MAC FROM ME” to neighbor PEs. The PE1 initiates clearing of the MAC addresses when its access side AC goes down. The peer PEs, PE2, PE3, and PE4, clear MAC addresses that are learned only from PE1.
- Scenario 2: When link Y, which is the AC of PE2 comes up, PE2 sends an LDP MAC withdrawal TLV message “FLUSH ALL MAC BUT ME” to neighbor PEs. The PEs clear the MAC addresses learned from the peer PEs, except those from the originating PE. In this example, PE2 is the originating PE.

Figure 2: MAC Address Withdrawal



Restrictions for Withdrawing Dynamic MAC Addresses Between Peer PE Routers

- MAC address withdrawal is not supported on the following:
  - Access Pseudowire (PW).
  - Hierarchical Virtual Private LAN Service (H-VPLS) network.
  - Network configured with BGP signaling and discovery.
- MAC withdraw relaying, the option to forward the received MAC withdraw messages, is not supported.

Configure MAC Address Withdrawal

Configure the following on PE1:

1. Create a bridge group and bridge domain.
2. Configure the bridge domain to withdraw the dynamically learned MAC addresses when the AC is down.
3. Associate the physical interface with the bridge domain.

```
/* Configuration on PE1 */
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)# withdraw state-down
Router(config-l2vpn-bg-bd-mac)# exit
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# commit
```

Running Configuration

```
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw state-down
   !
   interface HundredGigE0/0/0/0
   !
```

Disable MAC Address Withdrawal

MAC address withdrawal is enabled by default when the AC comes up. Configure the following on PE2, if you want to disable MAC address withdrawal.

```
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)# withdraw disable
Router(config-l2vpn-bg-bd-mac)# commit
```

Running Configuration

```
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw disable
   !
```

Verification

Run the **show l2vpn bridge-domain detail** command to verify the status of MAC address withdrawal.

The following example shows that MAC address withdrawal is enabled.

```
Router# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port down
```

The following example shows that MAC address withdrawal is disabled.

```
Router# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: disabled
    MAC withdraw sent on: bridge port up
```

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 8: 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.

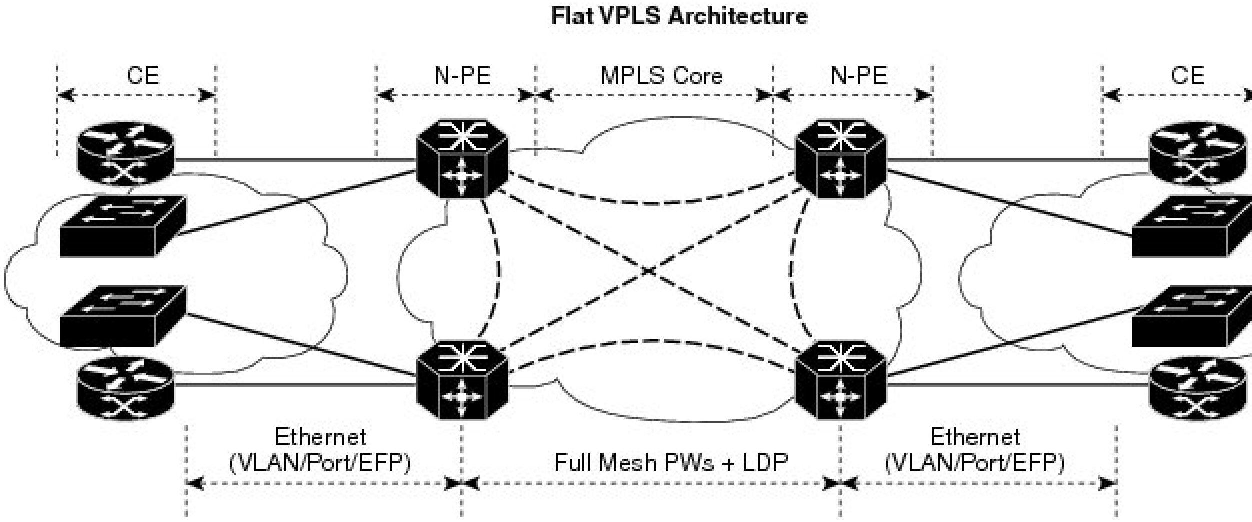
VPLS is a multipoint Layer 2 VPN technology that connects two or more provider edge (PE) devices using bridging. A bridge domain, which is the building block for multipoint bridging, is present on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment 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 3: 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
  !
!
```

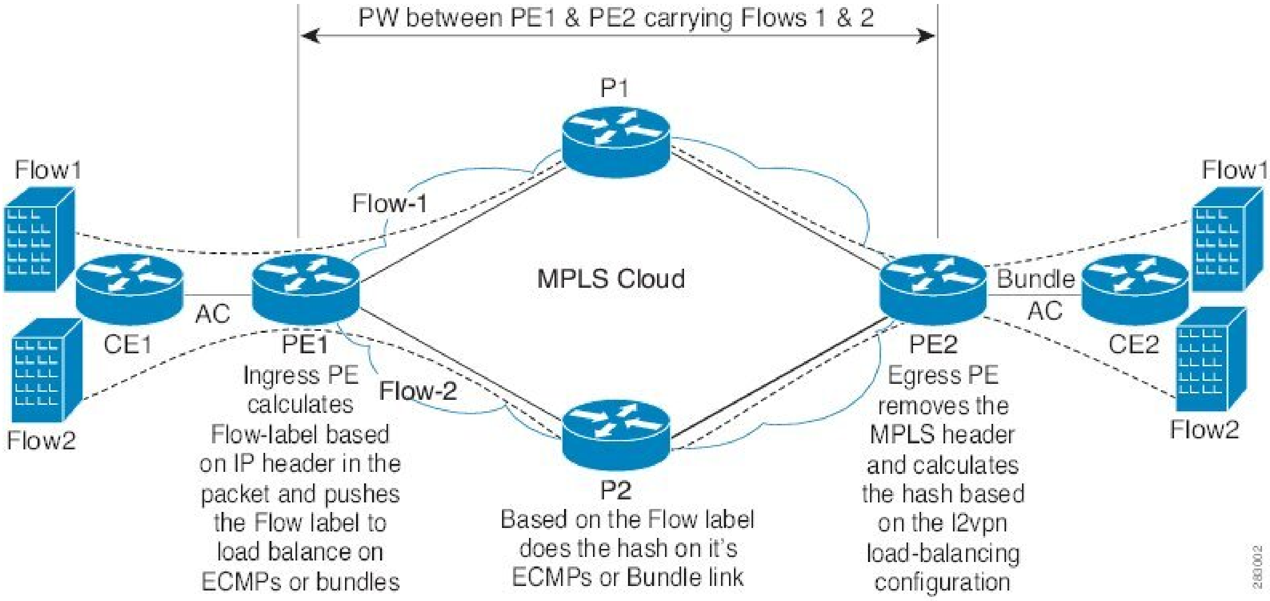
Load Balancing using Flow Aware Transport Pseudowire (FAT PW)

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 4: FAT PW with two flows distributing over ECMPs and Bundle-Links


PW between PE1 & PE2 carrying Flows 1 & 2
P1
Flow1
Flow-1
MPLS Cloud
Flow1
Bundle
AC
AC
CE1
PE1
PE2
CE2
Ingress PE calculates Flow-label based on IP header in the packet and pushes the Flow label to load balance on ECMPs or bundles
Flow-2
Flow2
P2
Based on the Flow label does the hash on it's ECMPs or Bundle link
Egress PE removes the MPLS header and calculates the hash based on the l2vpn load-balancing configuration
Flow2
280002

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
  !
!
```

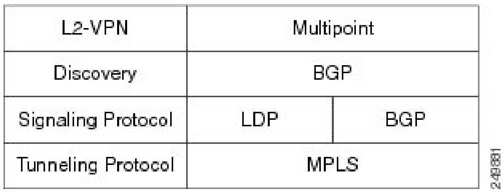
VPLS Discovery and Signaling

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 5: 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.

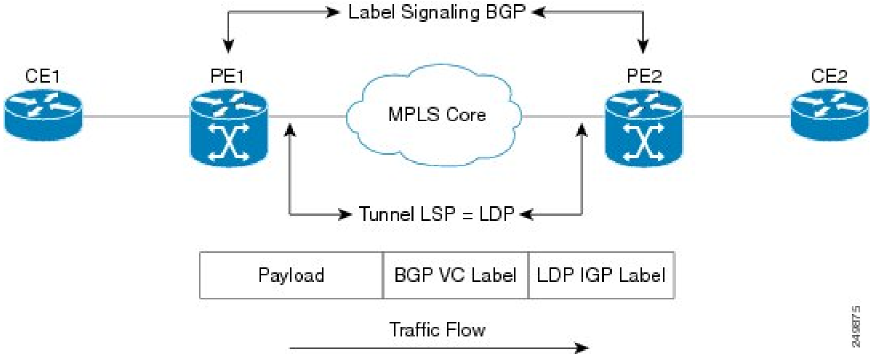
BGP-based VPLS autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS autodiscovery enables each VPLS PE router to discover the other provider edge (PE) routers that are part of the same VPLS domain. VPLS Autodiscovery also tracks when PE routers are added to or removed from the 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 6: Discovery and Signaling Attributes



The BGP signaling and autodiscovery scheme has the following components:

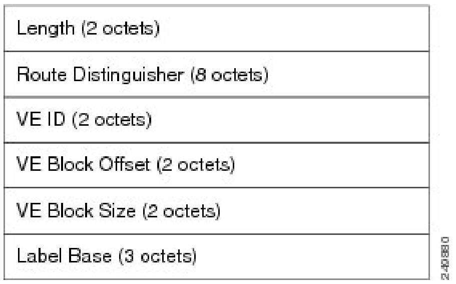
- A means for a PE to learn which remote PEs are members of a given VPLS. This process is known as autodiscovery.
- A means for a PE to learn the pseudowire label expected by a given remote PE for a given VPLS. This process is known as signaling.

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 7: NLRI Format

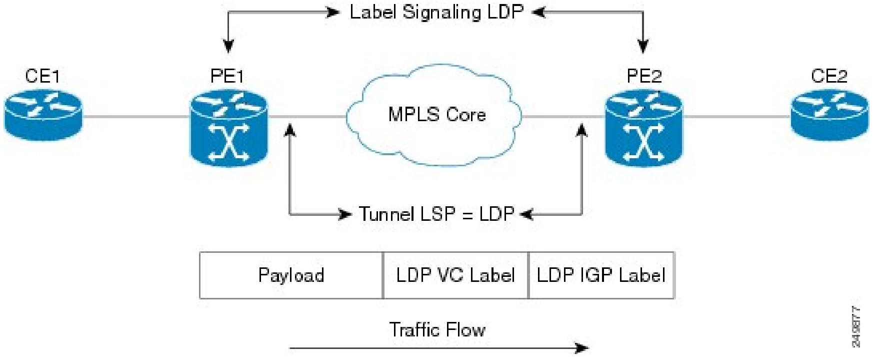


Length (2 octets)
Route Distinguisher (8 octets)
VE ID (2 octets)
VE Block Offset (2 octets)
VE Block Size (2 octets)
Label Base (3 octets)

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 8: 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.

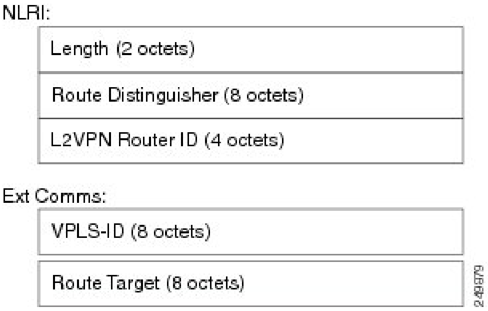
The LDP protocol is used to configure a pseudowire to all the other PE routers. FEC 129 is used for the signaling. The information carried by FEC 129 includes the VPLS ID, the Target Attachment Individual Identifier (TAII) and the Source Attachment Individual Identifier (SAII).

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 9: 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 9: 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.
- 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

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
CFM Hardware Offload	Release 7.9.1	CFM Hardware offloading allows to implement connectivity and fault monitoring for physical and bundle interfaces, using continuity check messages (CCM). This feature helps to detect network failure with short CCM intervals, which enables the router to recover from the failure without dropping the packets.

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

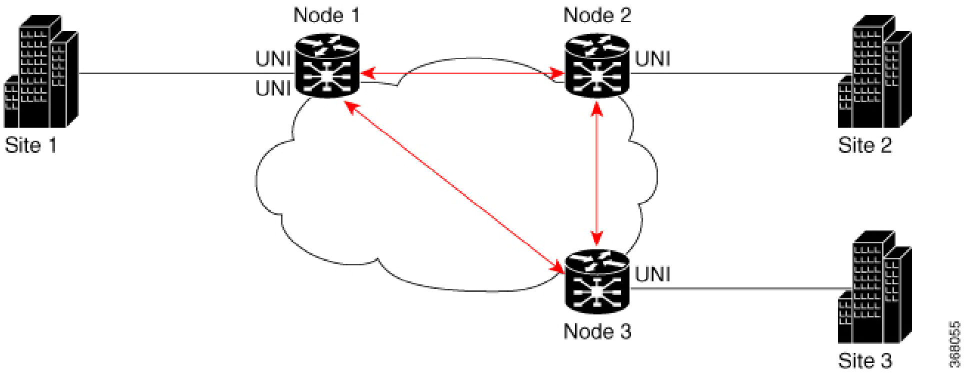
- Software offload—When CCMs are processed by the line card CPU, offload type is known as software offload. Software offload is supported only on bundle interface.
- Hardware offload—When CCMs are processed by network processor (NP), offload type is known as hardware offload.
- Non-offload—When CCMs 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 following table shows the supported CCM timers for the offload types:

Interface Type	Offload Type	Supported CCM Timers
Physical Interfaces and Subinterfaces	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
	Hardware offload	• 3.3 ms • 10 ms
Bundle Members	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
	Software offload	• 100 ms
Bundle Interfaces and Subinterfaces	Non-offload	• 1 sec • 10 sec • 1 min • 10 min
	Hardware offload	• 3.3 ms • 10 ms

Configure CFM on VPLS

Figure 10: 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 11: 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 12: 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.
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 13: 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 14: 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.

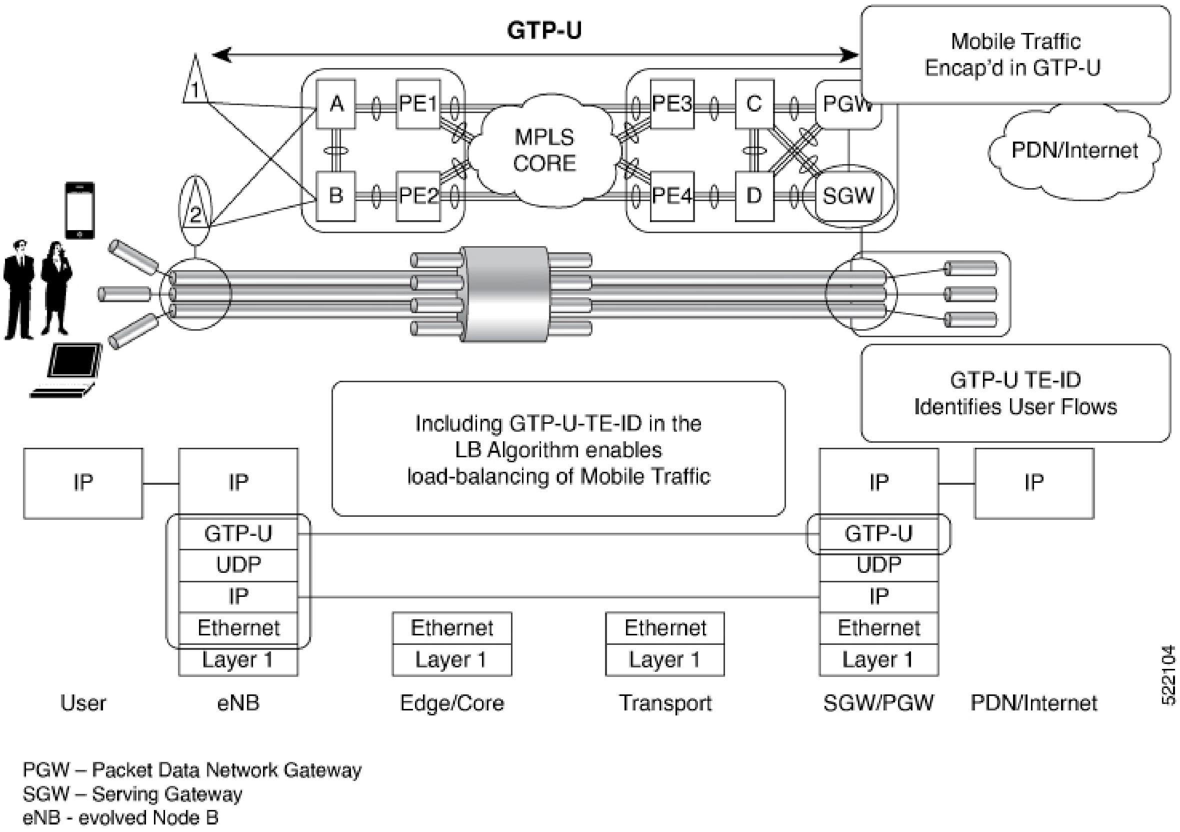
What is GTP?

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 11: Mobile Transport GTP-U Load-Balancing



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..

CHAPTER 8

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 15: 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.

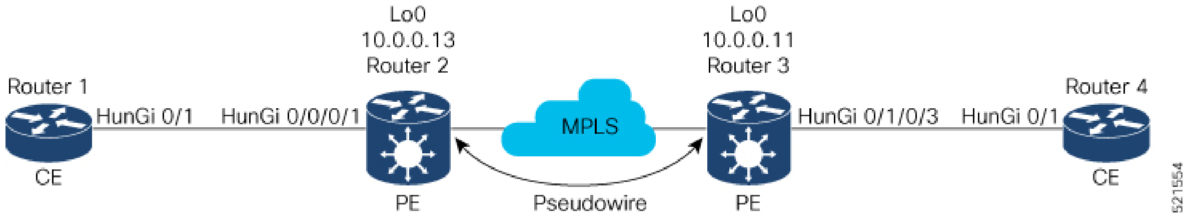
A pseudowire (PW) is a point-to-point connection between two provider edge (PE) devices which connects two attachment circuits (ACs). The two ACs connected at each PE are linked by a PW over the MPLS network, which is the MPLS PW.

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.

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.

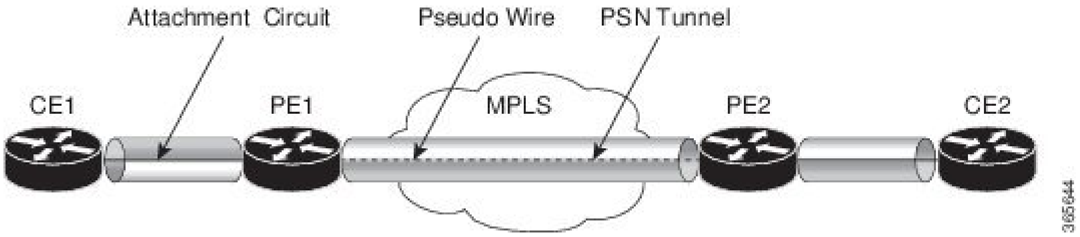
- The name of a cross-connect circuit is configured to identify a pair of PE routers and must be unique within the cross-connect group.
- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect circuit.
- A static virtual circuit local label is globally 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 12: 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

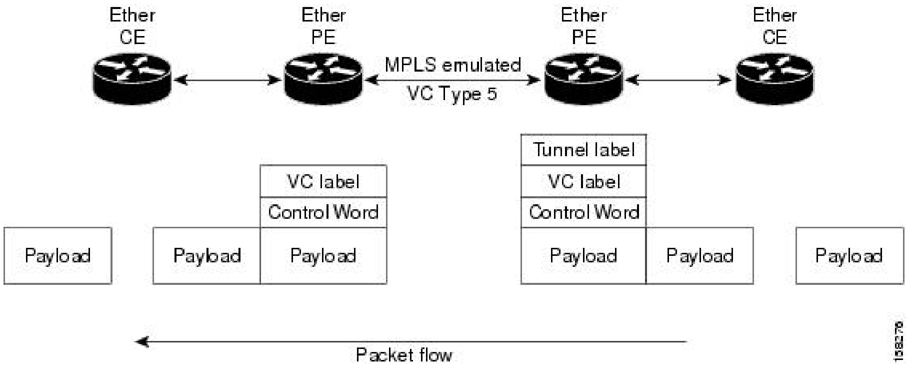
Table 16: Feature History Table

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 13: 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

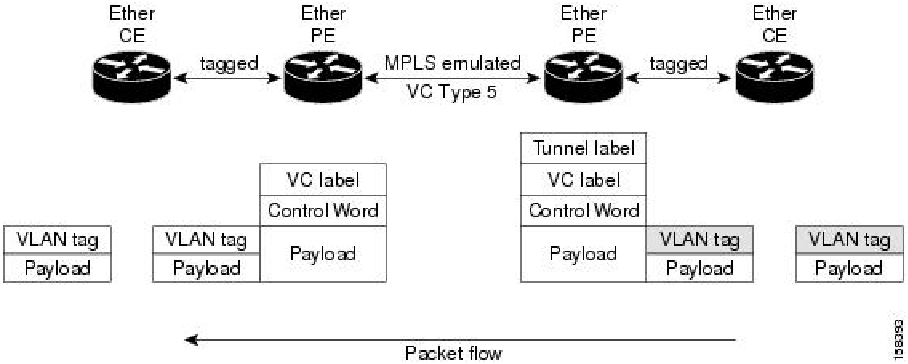
Table 17: Feature History Table

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 14: 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
```

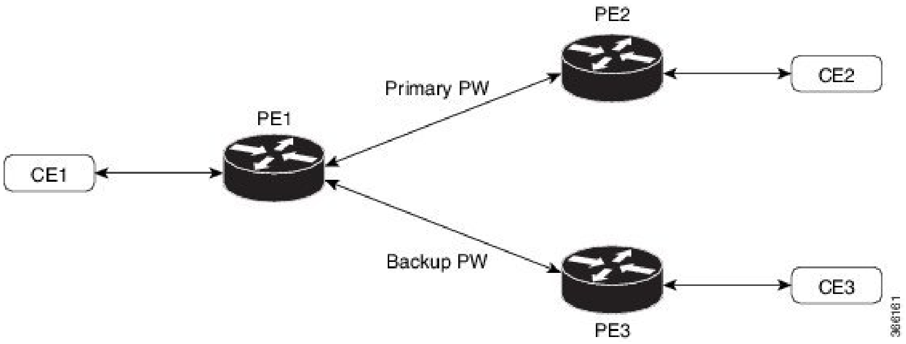
VLAN Passthrough Mode

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 15: 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 18: 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.

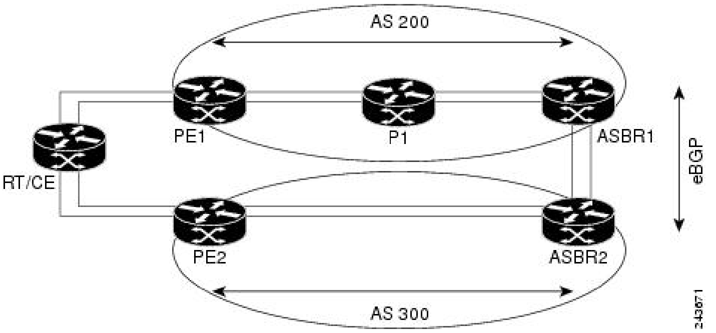
An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and uses a single, clearly defined routing protocol.

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 16: 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
```

Pseudowire Headend

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire Headend	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Pseudowire Headend (PWHE) is a virtual interface that allows termination of access PWs into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWHE enables integration of legacy Layer 2 services into packet-switched networks (PSNs) like IP or MPLS networks, so that users can integrate their older devices into newer networks without upgrading their hardware. This is possible because PWHE allows the termination or encapsulation of the frames from the attachment circuit into packets that can be transmitted over the PSN. * This feature is supported only on: • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Pseudowires (PWs) enable payloads to be transparently carried across IP/MPLS packet-switched networks (PSNs). PWs are regarded as simple and manageable lightweight tunnels for returning customer traffic into core networks.

Pseudowire Headend (PWHE) allows termination of access PWs into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWHE enables integration of legacy Layer 2 services into PSNs, so that users can integrate their older devices into newer networks without upgrading their hardware. This is possible because PWHE allows the termination of the frames from the attachment circuit into packets that can be transmitted over the PSN.

PWHE allows you to provision features such as QOS, access control lists (ACL), Layer 3 VPN on a per PWHE interface basis, on a Service Provider Edge (S-PE). In a typical Layer 2 VPN deployment, an attachment circuit (AC) is required to terminate the PWs. When PWHE is configured in a router, the PWHE interface emulates the behavior of an AC and terminates the PWs.

Note Only line cards and routers with the P100-based Silicon One ASIC support this feature.

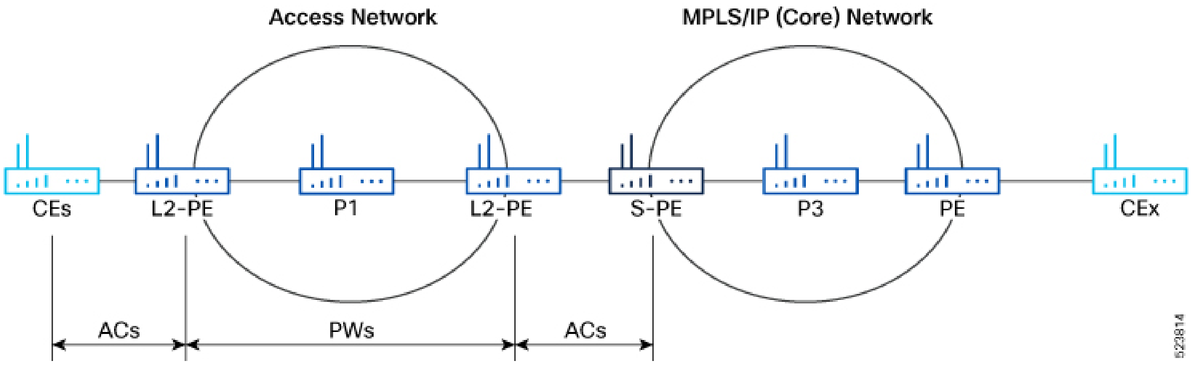
Benefits of PWHE

- Dissociates the customer facing interface (CFI) of the service PE from the underlying physical transport media of the access or aggregation network.
- Reduces capex in the access or aggregation network and service PE.
- Distributes and scales the customer facing Layer 2 user-network interfaces (UNI) set.
- Implements a uniform method of OAM functionality.
- Providers can extend or expand the Layer 3 service footprints.
- Provides a method of terminating customer traffic into a next generation network (NGN).

Topology

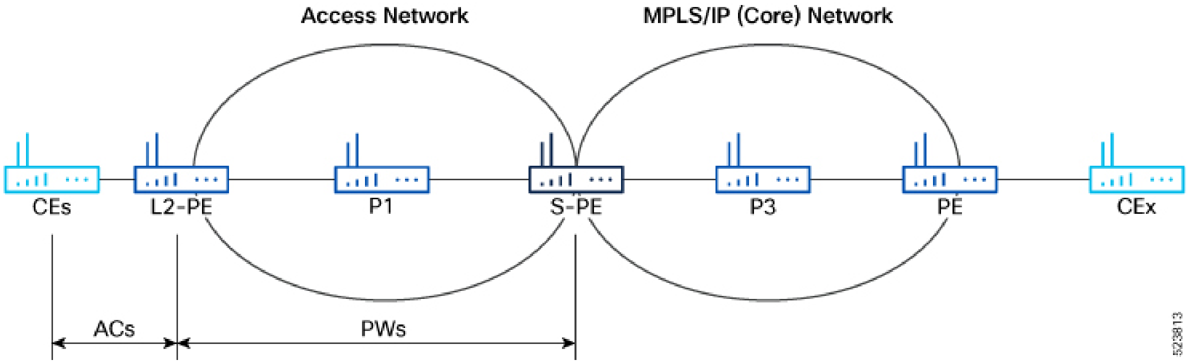
Consider a topology with Pseudowire network.

Figure 17: Pseudowire Network without PWHE



In this topology, the Layer 2 PE (L2-PE) from the access network is connected to the service provider edge (S-PE) through an AC. The PWs originating from L2-PE are terminated on the S-PE by the AC. This is a typical Layer 2 VPN deployment.

Figure 18: Pseudowire Network with PWHE



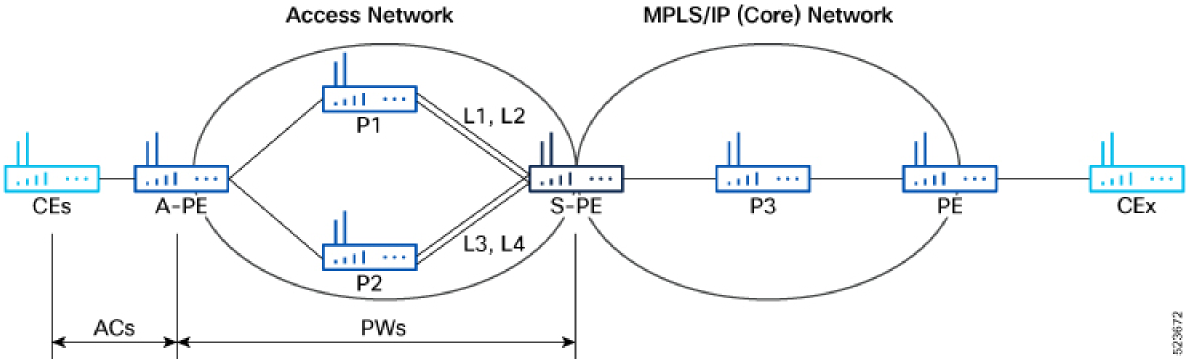
When you implement PWHE, the functionalities of L2-PE and S-PE are combined in S-PE. PWHE emulates the behavior of AC to terminate the PWs from the access network.

For PWHE cross-connect configuration, the interconnectivity between L2-PE and S-PE happens through BGP with RFC 3107 extension. The RFC 3107 allows BGP to distribute the MPLS labels along with IP prefixes. The customer network can avoid using an IGP to provide connectivity to the S-PE device, which is outside the customer's autonomous system.

PWs operate in bridged interworking mode with Virtual Circuit (VC) type 5. The VC type indicates the mode in which the packets are processed. VC type 5 is used for Ethernet port mode, where the PWs carry customer Ethernet frames (tagged or untagged) with IP payload. Thus, an S-PE device must perform ARP resolution for customer IP addresses learned over the PWHE.

Consider a topology with PWHE deployment where the access network consists of multiple provider routers.

Figure 19: PWHE Deployment



In this topology, there are multiple CEs connected to access PE (A-PE) with each CE connected by one link.

- There are two provider routers, P1 and P2, available between A-PE and S-PE in the access network.
- The S-PE is connected to P1 using links L1 and L2. The links L1 and L2 are connected to two different line cards on P1 and S-PE.
- The S-PE is connected to P2 using links L3 and L4. The links L3 and L4 are connected to two different line cards on P2 and S-PE.
- For each CE-A-PE link, a cross-connect (AC-PW) is configured on the A-PE.

While configuring mulitple PWs, group the PWs under a generic interface list (GIL). When you modify the configuration of a GIL, the changes are applied to all the PW interfaces associated with the GIL. For more information on GIL, see Generic Interface List, on page 86.

Traffic Flow Types on PWHE Interfaces

There are two types of traffic flow involved in the PWHE interfaces.

- **PWHE Decapsulation/Disposition Flow**: On traffic flowing from access side to core, PWHE ingress features would be executed on the PWHE Access facing Line card.
- **PWHE Encapsulation/Imposition Flow**: On traffic flowing from core to access side, PWHE egress features would be executed on the PWHE Access facing Line card.

PWHE Decapsulation

The packets from access side to core side are decapsulated and the ingress features are executed.

1. The outer L2 header or VLAN tag is removed from the packet that arrives at PWHE router.
2. The PW label is removed from the packet.
3. The inner L2 is removed from the packet.
4. The packet is reconstructed with the transport label, service label, and L2 header.
5. The reconstructed packet is delivered to the core network.

PWHE Encapsulation

The packets from core side to access side are encapsulated and the egress features are executed.

- The inner L2 header is encapsulated.
- The packet is reconstructed with the PW label, transport labels, and outer L2 header.
- The reconstructed packet is delivered to the access network.

Generic Interface List

A generic interface list (GIL) is a list of physical or bundle interfaces used in a PWHE connection.

The GIL supports only main interfaces, and not subinterfaces. The GIL is bi-directional and restricts both receive and transmit interfaces on access-facing line cards. The GIL has no impact on the core-facing side.

A GIL is used to limit the resources allocated for a PWHE interface to the set of interfaces specified in the list.

Only the S-PE is aware of the GIL and expects that the PWHE packets arrive on only line cards with GIL members on it. If packets arrive at the line card without GIL members on it, they are dropped.

Restrictions for Pseudowire Headend

- The following are not supported on PWHE

  - ISIS as IGP for access core
  - PW classVC label 4
  - Segment Routing Traffic Engineering (SR-TE) in the access core for Layer 2 VPN
  - TE tunnel as preferred path in access core for Layer 2 VPN
  - Traffic with Internet Mix (IMIX)
  - The commands **load-balancing flow src-dst-ip** and **flow-label both**
  - PWHE load balancing by VC label or by Flow-Aware Transport (FAT)
  - EVPN mulithoming mode
  - PWHE MTU

- The load balancing hashing is done only by PWHE link numbers.
- When a packet arrives at PWHE Layer 3 subinterface, the software aggregate count is done on PWHE main interface.
- It is recommended not to use mixed-mode Egress Traffic Management (ETM), a combination of ETM and non-ETM members in a GIL.
- It is recommended not to use the ECMP path list as a superset of GIL interfaces.
- If you have configured other features like QoS and ACL on GIL, they are applicable as follows:
  - For PWHE traffic received on GIL, all the features configured on the PWHE interface are applicable.
  - For other traffic received on GIL, all the features configured on the GIL interface are applicable.
- You can attach ACL in PWHE interface for both ingress and egress, for IPv4 and IPv6.
- You can attach hybrid-ACL (Level 2) in PWHE interface for only ingress, for IPv4 and IPv6.

Configure Pseudowire Headend

Prerequisites

Consider the following guidelines while configuring PWHE:

- The generic interface list members must be the superset of the ECMP path list to the Access Provider Edge (A-PE).
- Only eight generic interface lists are supported per A-PE neighbor address.
- Eight Layer 3 links per generic interface list are supported.
- Only PW-Ether interfaces can be configured as PWHE L2 or L3 subinterfaces.
- Cross-connects that contain PW-Ether main interfaces can be configured as VC-type 5.
- PW-Ether interfaces and subinterfaces can be configured with both IPv4 and IPv6. To encapsulate and transport IPv4 or IPv6 frames over a pseudowire, the packet-switched network must be capable of routing IPv4 and IPv6 packets.

- Pseudowire redundancy, preferred path, local switching or L2TP are not supported for cross-connects configured with PWHE.
- The TE and LDP applications work on physical interfaces and therefore do not allow PWHE configuration.
- Address family, CDP, and MPLS configurations are not allowed on PWHE interfaces.
- For PWHE, eBGP, static routes, OSPF, and ISIS are supported with both IPv4 and IPv6. Routing Information Protocol (RIP) is only supported with IPv4 and not with IPv6.
- You must attach a different generic interface list for PW-Ether interfaces with different remote neighbors. Hence, you must create a separate and dedicated generic interface list for each remote neighbor or peer whose remote neighbors are different routers or devices. For example, a unique generic interface list needs to be configured for each Access Provider Edge that the Access Provider Edge peers with. The generic interface list may have the same set of outgoing interfaces.

Configuration Example

A-PE Configuration

Configure the A-PE with PWHE cross-connect to include the L2 interface and PW towards S-PE. Cross-connect or xconnect is used to establish connection between the Access Provider and Service Provider Edges.

```
/* Configure L2 interface */
Router(config)# interface hundredGigE 0/1/0/3
Router(config-if)# l2transport
Router(config-if-l2)# root
```

```
/* Configure PWHE cross-connect */

Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface hundredGigE 0/1/0/3
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 10.1.1.1 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

S-PE Configuration

Configure the S-PE:

- Create generic interface list (GIL).
- Configure pw-ether interface and attach the GIL to the interface.
- Configure cross-connect to include pw-ether interface. Use the **transport-mode ethernet** command to transport the PW traffic through Ethernet.

```
/* Create GIL */
Router(config)# generic-interface-list txlist
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

```
/* Configure pw-ether interface and attach GIL */
Router(config)# interface pw-ether1
Router(config-if)# ipv4 address 10.1.1.1/24
Router(config-if)# ipv6 address 5000::2/64
Router(config-if)# attach generic-interface-list txlist
Router(config-if)# commit

/* Configure cross-connect to include pw-ether interface */
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface pw-ether1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.2.2.2 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

L2 Subinterface Configuration

The following example shows how to configure PWHE on L2 subinterface.

A-PE Configuration

Configure L2 subinterface.

```
/* Configure L2 subinterface */
Router(config)# interface hundredGigE 0/1/0/3.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# rewrite ingress tag pop 1 symmetric
```

Note There is no need to configure cross-connect for the subinterface, as the main PWHE interface configuration is applied to the subinterface.

S-PE Configuration

Configure the L2 subinterface and add the subinterface.

```
/* Configure L2 subinterface */
Router(config)# interface PW-Ether1.2 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric

/* Configure cross-connect and assign the L2 subinterface  */
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
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 pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface pw-ether1.2
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.2.2.2 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

L3 Subinterface Configuration

The following example shows how to configure PWHE on L3 subinterface. Except the A-PE, repeat the other configurations as described for L2 interface.

A-PE Configuration

```
/* Configure L3 subinterface */
Router(config)# interface pw-ether 1.1
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ipv4 address 10.1.1.1/24
Router(config-subif)# commit
```

Note There is no need to configure cross-connect for the subinterface, as the main PWHE interface configuration is applied to the subinterface.

Running Configuration

```
/* A-PE Configuration */
interface hundredGigE 0/1/0/3
  l2transport
!
l2vpn
   pw-class pwhe_port_vc5
   encapsulation mpls
  xconnect group pw-he
p2p pw-ether1
interface hundredGigE 0/1/0/3
neighbor ipv4 10.1.1.1 pw-id 6
pw-class pwhe_port_vc5
```

```
/* S-PE Configuration */
generic-interface-list txlist
  interface hundredGigE 0/1/0/1
  interface hundredGigE 0/1/0/2
!
interface PW-Ether1
  ipv4 address 10.1.1.1/24
  ipv6 address 5000::2/64
  attach generic-interface-list txlist
!
l2vpn
  pw-class pwhe_port_vc5
     encapsulation mpls
      transport-mode ethernet
  xconnect group pw-he
     p2p pw-ether1
       interface PW-Ether1
       neighbor ipv4 10.2.2.2 pw-id 6
         pw-class pwhe_port_vc5
```

```
/* A-PE Configuration for L2 Subinterface */
interface hundredGigE 0/1/0/3.2 l2transport
   encapsulation dot1q 2
   rewrite ingress tag pop 1 symmetric
```

```
/* S-PE Configuration for L2 Subinterface */
l2vpn
    pw-class pwhe_port_vc5
        encapsulation mpls
        transport-mode vlan
    !
    xconnect group pw-he
        p2p pw-ether1
            interface pw-ether1.2
            neighbor ipv4 10.2.2.2 pw-id 6
            pw-class pwhe_port_vc5


/* A-PE Configuration for L3 subinterface */
interface pw-ether 1.1
    encapsulation dot1q 1
    ipv4 address 10.1.1.1/24
```

Verification

The following output shows the details of GIL.

```
Router# show generic-interface-list idb name txlist

GIL name: txlist, ifhandle: 0xf000014
  State: Down Immediate
  Members:
    HundredGigE0/1/0/2
    HundredGigE0/1/0/1
```

The following output shows the status of xconnect group with PWHE Up.

```
Router# show l2vpn xconnect group pw-he xc-name pw-ether1
Mon Mar 11 21:26:48.281 EDT
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                  Segment 1       Segment 2
Group     Name        ST  Description ST Description              ST
------------------------ ----------------------------- -----------------------------
pw-he pw-ether1       UP PE1          UP EVPN 1,1,102.102.102.102 UP
----------------------------------------------------------------------------------------
```

The following output shows the status of PWHE interface with PWHE Up.

```
Router# show l2vpn pwhe interface pw-ether 1 detail

Interface: PW-Ether1 Interface State: Up, Admin state: Up
Interface handle 0xf000054
MTU: 1514
BW: 10000 Kbit
Interface MAC addresses: 1859.f57d.0008
Label: 24041
Internal ID: None
L2-overhead: 0
VC-type: 5
CW: Y
Hash: 0xf5f5 [Success]
```

Traffic Mirroring on PWHE

Table 20: Feature History Table

Feature Name	Release Information	Feature Description

Traffic Mirroring on PWHE	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) This feature allows you to perform a detailed inspection and analysis of layer 2 network traffic passing through a set of ethernet interfaces without interrupting the flow of traffic. You can copy or mirror the network packets that pass through a specific source interface within a layer 2 VPN, allowing these packets to be redirected to a predetermined destination interface. *This feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Traffic mirroring, also known as Switched Port Analyzer (SPAN) is a traffic monitoring and analysis tool that enables a user to monitor the network traffic passing through a set of ethernet interfaces.

Cisco IOS XR Release 24.3.1 introduces traffic mirroring on PWHE. This allows the mirroring of packets that pass through a source interface to a specified destination interface. The destination interface may then be attached to a network analyzer for debugging. For more information about the Traffic Mirroring, see the *Configuring Traffic Mirroring* chapter in the *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*. For complete command reference for this feature, see the *Traffic Mirroring Commands* chapter in the *Interface and Hardware Component Command Reference for Cisco 8000 Series Routers*.

SPAN feature support on PWHE

PWHE supports these SPAN features:

- ERSPAN: This feature allows you to monitor traffic over an IP network to a remote monitoring device. You can analyze the traffic from a switch that is not locally connected to the monitoring device.
- Security ACL with or without UDF: This feature allows you to use the Access Control Lists (ACLs) to filter network traffic based on certain criteria. UDFs can be used within ACLs to create more granular and customized filtering rules.
- Forward drop mirroring: This feature allows you to copy or mirror packets that are dropped during the forwarding process at the router ingress to a configured destination. These mirrored packets can be captured and analyzed using network monitoring tools. The analysis of dropped packets helps you understand the types of traffic that are blocked, analyze potential security threats, troubleshoot, and optimize network performance.
- Truncation: This feature allows you to capture only a portion of each packet using a monitoring tool. This is useful for reducing the amount of data that needs to be processed and stored during network analysis.
- SPAN to file: This feature allows you to captured traffic to be written directly to a file for later analysis. This is useful for archiving data or for situations where real-time analysis is not required.
- Buffer drop mirroring: This feature allows you to mirrors packets that are dropped at the ingress buffer of a switch. This is useful for diagnosing network congestion problems.

For configuring traffic mirroring on PWHE, see the Configuring Traffic Mirroring Features on the PWHE Interface, on page 93.

Limitations and Restrictions for Traffic Mirroring on PWHE

These are the limitations and restrictions of traffic mirroring on PWHE:

- PWHE don't support these SPAN features:
  - Egress SPAN
  - Local SPAN
  - Span ACL
  - Mixed mode SPAN
  - Multi SPAN ACL
- The truncation supports capturing packet sizes up to 176 bytes for IPv4 traffic and 196 bytes for IPv6 traffic.
- Linecards and fixed routers with Q200 and P100 based Silicon One ASICs support buffer-drop mirroring.

Configuring Traffic Mirroring Features on the PWHE Interface

You can configure any of these SPAN features on the PWHE interface as per your specific network monitoring and troubleshooting requirements:

- ERSPAN. Refer to Configuring ERSPAN on the PWHE Interface, on page 93.
- Security ACL. Refer to Configuring Security ACL on the PWHE Interface, on page 96.
- Forward drop mirroring. Refer to Configuring Forward Drop Mirroring on the PWHE Interface, on page 98.
- Truncation. Refer to Configuring Truncation on the PWHE Interface, on page 100.
- SPAN to file. Refer to Configuring Span to File on the PWHE Interface, on page 101.
- Buffer drop mirroring. Refer to Configuring Buffer Drop Mirroring on the PWHE Interface, on page 102.

Configuring ERSPAN on the PWHE Interface

Perform these steps to configure the ERSPAN on the PWHE interface:

Step 1 Configure a Generic Interface List (GIL) with the **generic-interface-list** command to include multiple interfaces within a generic interface.

Example:

```
Router# configure
Router(config)# generic-interface-list gil
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

Step 2 Configure a pseudowire interface with the **interface pw-ether** command and attach a GIL to the interface to encapsulate and transport the mirrored traffic.

Example:

```
Router# configure
Router(config)# interface pw-ether1
Router(config-if)# mac-address aa39.3362.3007
Router(config-if)# ipv4 address 200.0.0.1 255.255.255.0
Router(config-if)# attach generic-interface-list gi1
```

Step 3 Configure a traffic monitoring session with the **monitor-session** command to monitor the inbound traffic.

Example:

```
Router(config-if)# monitor-session mysession ethernet direction rx-only
Router(config-if)# commit
```

Step 4 View the running configuration to verify the configuration that you have configured.

Example:

```
/* Create GIL */
generic-interface-list gi1
    interface hundredGigE 0/1/0/1
    interface hundredGigE 0/1/0/2
!
/* Configure pw-ether interface and attach GIL */
interface PW-Ether1
    mac-address aa39.3362.3007
    ipv4 address 200.0.0.1 255.255.255.0
    attach generic-interface-list gi1

/* Apply traffic monitoring session to the pw-ether interface*/
    monitor-session mysession ethernet direction rx-only
!
```

Step 5 Verify the ERSPAN configuration on the PWHE interface with the **show monitor-session <session_name> status** command and **show monitor-session <session_name> status internal** command for session statistics.

In the following example, you can verify the ERSPAN configuration on the PWHE interface with the **show monitor-session <session_name> status** command.

```
Router# show monitor-session status
Monitor-session mon1
Destination interface tunnel-ip1
================================================================================
Source Interface            Dir         Status
---------------------      ----   ---------------------------------------------------
Hu0/1/0/14                  Rx     Operational
Hu0/1/0/15.100              Rx     Operational
BE1                         Rx     Operational
BE1.1                       Rx     Operational
```

In the following example, you can verify the ERSPAN configuration on the PWHE interface with the **show monitor-session <session_name> status internal** command.

```
Router# show monitor-session status internal
Information from SPAN Manager and MA on all nodes:
Monitor-session mon1 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface <>(0x00800190)
Last error: Success
0/1/CPU0: Destination interface <>(0x00800190)
0/RP0/CPU0: Destination interface <>(0x00800190)
```

```
Information from SPAN EA on all nodes:
Monitor-session 0x00000001 (Ethernet)
0/1/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/1/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 32
Monitor Session Bytes: 4024
0/2/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/2/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 0
Monitor Session Bytes: 0
```

Step 6 Capture the packets using a traffic generator tool.

Step 7 Verify that the captured packet is a GRE packet and should match the original sent packet encapsulated with a GRE header.

Example:

The following example of a decoded ERSPAN captured packet displays that the original packet is encapsulated with a GRE header:

```
/*Captured GRE packet*/
Frame 13: 1330 bytes on wire (10640 bits), 1330 bytes captured (10640 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 30, 2023 16:06:03.167332961 EST
    UTC Arrival Time: Nov 30, 2023 21:06:03.167332961 UTC
    Epoch Arrival Time: 1701378363.167332961
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000001910 seconds]
    [Time delta from previous displayed frame: 0.000001910 seconds]
    [Time since reference or first frame: 0.500001960 seconds]
    Frame Number: 13
    Frame Length: 1330 bytes (10640 bits)
    Capture Length: 1330 bytes (10640 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:gre:erspan:eth:ethertype:ip:data]

/*GRE header*/
Ethernet II, Src: Cisco_75:50:dc (34:88:18:75:50:dc), Dst: Cisco_00:00:01 (00:12:01:00:00:01)
    Destination: Cisco_00:00:01 (00:12:01:00:00:01)
    Source: Cisco_75:50:dc (34:88:18:75:50:dc)
    Type: IPv4 (0x0800)
    Frame check sequence: 0x41b15e2b [unverified]
    [FCS Status: Unverified]
Internet Protocol Version 4, Src: 7.0.0.1, Dst: 7.0.0.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1312
    Identification: 0x0000 (0)
    010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 254
    Protocol: Generic Routing Encapsulation (47)
    Header Checksum: 0x69ac [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 7.0.0.1
    Destination Address: 7.0.0.2
Generic Routing Encapsulation (ERSPAN)
Encapsulated Remote Switch Packet ANalysis Type II
```

```
/*Original packet*/
Ethernet II, Src: Cisco_00:ee:00 (28:af:fd:00:ee:00), Dst: Cisco_75:50:ec (34:88:18:75:50:ec)
    Destination: Cisco_75:50:ec (34:88:18:75:50:ec)
    Source: Cisco_00:ee:00 (28:af:fd:00:ee:00)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 13.0.0.2, Dst: 7.0.0.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1262
    Identification: 0x0000 (0)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 63
    Protocol: any host internal protocol (61)
    Header Checksum: 0x62d0 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 13.0.0.2
    Destination Address: 7.0.0.2
Data (1242 bytes)
    Data [truncated]: ecc0d6806f768300497869606000000001011121 39c1d0b9904be1a1b1c1d1e1f202
                      12223242526272829 2a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142
                      434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f606162636
                      465666768696a6b6c6d6
    [Length: 1242]
```

Configuring Security ACL on the PWHE Interface

Perform these steps to configure the security ACL on the PWHE interface:

Step 1 Configure a Generic Interface List (GIL) with the **generic-interface-list** command to include multiple interfaces within a generic interface.

Example:

```
Router# configure
Router(config)# generic-interface-list gi1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

Step 2 Configure a pseudowire interface with the **interface pw-ether** command and attach a GIL to the interface to encapsulate and transport the mirrored traffic.

Example:

```
Router# configure
Router(config)# interface pw-ether1
Router(config-if)# mac-address aa39.3362.3007
Router(config-if)# ipv4 address 200.0.0.1 255.255.255.0
Router(config-if)# attach generic-interface-list gi1
Router(config-if)# commit
```

Step 3 Configure security ACL with the **ipv4 access-list span-acl** command to create a specific ACL that can be applied to monitor and control traffic on the PWHE interface.

Example:

```
Router# configure
Router(config)# ipv4 access-list span-acl1
Router(config)# 10 permit ipv4 any host 54.0.0.2 capture
Router(config)# commit
```

Step 4 Apply the security ACL session to the configured pseudowire interface with the **monitor-session** command to monitor inbound traffic and apply an ACL to filter that traffic.

Example:

```
Router# configure
Router(config)# interface pw-ether1
Router(config-if)# monitor-session span0 ethernet direction rx-only ac
Router(config-if)# ipv4 access-group span-acl0 ingress
Router(config-if)# commit
```

Step 5 View the running configuration to verify the configuration that you have configured.

Example:

```
/* Create GIL */
generic-interface-list gi1
    interface hundredGigE 0/1/0/1
    interface hundredGigE 0/1/0/2
!
/* Configure pw-ether interface and attach GIL */
interface PW-Ether1
    mac-address aa39.3362.3007
    ipv4 address 200.0.0.1 255.255.255.0
    attach generic-interface-list gi1
!
/*Configure ACL*/
ipv4 access-list span-acl1
    10 permit ipv4 any host 54.0.0.2 capture

/* Apply ACL session to the pw-ether interface*/
    interface pw-ether1
    monitor-session span0 ethernet direction rx-only ac
    ipv4 access-group span-acl0 ingress
!
```

Step 6 Verify the security ACL configuration on the PWHE interface with the **show monitor-session <session-name> status** command and **show monitor-session <session-name> status internal** command for session statistics.

In the following example, you can verify the security ACL configuration on the PWHE interface using the **show monitor-session <session-name> status** command.

```
Router# show monitor-session status
Monitor-session mon1
Destination interface tunnel-ip1
================================================================================
Source Interface            Dir         Status
---------------------      ----   ----------------------------------------------------
Hu0/1/0/14                  Rx    Operational
Hu0/1/0/15.100              Rx    Operational
BE1                         Rx    Operational
BE1.1                       Rx    Operational
```

In the following example, you can verify the security ACL configuration on the PWHE interface with the **show monitor-session <session-name> status internal** command.

```
Router# show monitor-session status internal
Thu Aug 13 20:05:23.478 UTC
Information from SPAN Manager and MA on all nodes:
```

```
Monitor-session mon1 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface <>(0x00800190)
Last error: Success
0/1/CPU0: Destination interface <>(0x00800190)
0/RP0/CPU0: Destination interface <>(0x00800190)
Information from SPAN EA on all nodes:
Monitor-session 0x00000001 (Ethernet)
0/1/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/1/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 32
Monitor Session Bytes: 4024
0/2/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/2/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 0
Monitor Session Bytes: 0
```

Configuring Forward Drop Mirroring on the PWHE Interface

Perform these steps to configure the forward drop mirroring on the PWHE interface:

Step 1 Configure a traffic monitoring session with the **monitor-session** command.

ERSPAN and span to file features supports the forward drop mirroring. You can configure a traffic monitoring session with the ERSPAN or span to file feature to enable forward drop mirroring.

- Configure a traffic monitoring session with span to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
Router(config-mon)# commit
```

- Configure a traffic monitoring session with ERSPAN to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination tunnel-ip1
```

Step 2 Capture packet drops with the **forward-drop** or **drops packet-processing rx** command to capture packets at all ports on the network forwarding module.

Example:

```
Router(config-mon)# forward-drop rx
```

Step 3 View the running configuration to verify the configuration that you have configured.

Example:

```
/*configure a traffic monitoring session*/
monitor-session mysession ethernet
    destination file
    forward-drop rx
!
```

Step 4 Verify the forward drop mirroring configuration on the PWHE interface with the **show spp node-counters** command.

In the following example, you can verify the froward drop mirroring configuration on the PWHE interface.

```
Router# show spp node-counters
Tue Feb 27 21:12:49.886 UTC
0/1/CPU0:
socket/rx
                ether raw pkts:              10
                low que accept:              10
           sent to XR classify:              10
-------------------------------
socket/tx
                       ce pkts:              20
        SW padding vector used:              20
-------------------------------
device/classify
      forwarded to spp clients:              10
 forwarded NPU packet to NetIO:              10
            L3 Route not found:              10
-------------------------------
client/inject
        pkts injected into spp:              10
  NetIO->CPU injected into spp:              10
  NetIO->CPU PKT IPV4_PREROUTE:              10
-------------------------------
pd_span_drop
                     SPAN drop:              10
-------------------------------
client/punt
              punted to client:              10
-------------------------------
```

Step 5 Perform these steps to capture and verify packets.

- Perform these steps to capture and verify forward drop mirroring using span to file feature:

a) Enable capturing the forward drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

b) Stop capturing the forward drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

c) Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

d) Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

e) Verify that the captured packet matches the original packet.

- Perform these steps to capture and verify forward drop mirroring using ERSPAN feature:

a) Capture the packets with a traffic generator tool.
b) Verify that the captured packet is a GRE packet and should match the original sent packet encapsulated with a GRE header.

Configuring Truncation on the PWHE Interface

Perform these steps to configure the truncation on the PWHE interface:

Step 1 Configure a traffic monitoring session with the **monitor-session** command to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
```

Step 2 Truncate the mirrored packets to a specific length with the **mirror first** command to capture only the required information for monitoring and troubleshooting.

Example:

```
Router(config-mon)# mirror first 128
 Router(config-mon)# commit
```

Step 3 Configure the traffic monitoring session with the **monitor-session** command to monitor the inbound traffic.

Example:

```
Router(config-if)# monitor-session mysession ethernet direction rx-only
```

Step 4 View the running configuration to verify the configuration that you have configured.

Example:

```
/*configure a traffic monitoring session*/
monitor-session mysession ethernet
    destination file

/*Truncate the mirrored packets to a specific length*/
    mirror first 128
!
/*Configure the traffic monitoring session to monitor the inbound traffic*/
monitor-session mysession ethernet direction rx-only
```

Step 5 Verify the truncation configuration on the PWHE interface with the **show spp node-counters** command.

In the following example, you can verify the truncation configuration on the PWHE interface.

```
Router# show spp node-counters
Tue Feb 27 21:12:49.886 UTC
0/1/CPU0:
socket/rx
                ether raw pkts:              10
                low que accept:              10
           sent to XR classify:              10
```

```
-------------------------------
socket/tx
                       ce pkts:              20
        SW padding vector used:              20
-------------------------------
device/classify
      forwarded to spp clients:              10
 forwarded NPU packet to NetIO:              10
            L3 Route not found:              10
-------------------------------
client/inject
        pkts injected into spp:              10
  NetIO->CPU injected into spp:              10
  NetIO->CPU PKT IPV4_PREROUTE:              10
-------------------------------
pd_span_drop
                     SPAN drop:              10
-------------------------------
client/punt
              punted to client:              10
-------------------------------
```

Step 6 Enable capturing the truncation packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

Step 7 Stop capturing the truncation packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

Step 8 Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

Step 9 Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

Configuring Span to File on the PWHE Interface

Perform these steps to configure the span to file on the PWHE interface:

Step 1 Configure a traffic monitoring session with the **monitor-session** command to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
Router(config-mon)# commit
```

Step 2 Configure the traffic monitoring session with the **monitor-session** command to monitor the inbound traffic.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet direction rx-only
```

Step 3 View the running configuration to verify the configuration that you have configured.

Example:

```
Router# show running-config interface hundredGigE 0/1/0/0
interface HundredGigE0/1/0/0
 ipv4 address 10.0.0.1 255.255.255.0
 monitor-session mysession ethernet direction rx-only
```

Step 4 Verify the the span to file configuration on the PWHE interface with the **show spp node-counter** command.

In the following example, you can verify the span to file configuration on the PWHE interface.

```
Router# show spp node-counters location 0/5/CPU0  | i SPAN
                  SPAN to File:          299846
```

Step 5 Enable capturing the span to file packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

Step 6 Stop capturing the span to file packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

Step 7 Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

Step 8 Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

Configuring Buffer Drop Mirroring on the PWHE Interface

Only the linecards and fixed routers with Q200 and P100 based silicon one ASICs support buffer drop mirroring.

Perform these steps to configure the buffer drop (TM drop) mirroring on the PWHE interface.

Step 1 Configure a traffic monitoring session with the **monitor-session** command.

ERSPAN and span to file features supports the buffer drop mirroring. You can configure a traffic monitoring session for buffer drop mirroring with the ERSPAN or span to file feature.

- Configure a traffic monitoring session with span to file to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
Router(config-mon)# commit
```

- Configure a traffic monitoring session with ERSPAN to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination tunnel-ip1
```

Step 2 Capture packet drops with the **tm-drop** or **drops traffic-management rx** command to capture packets at all ports on the network forwarding module.

Example:

```
Router(config-mon)# tm-drop rx
```

Step 3 View the running configuration to verify the configuration that you have configured.

Example:

```
/*configure a traffic monitoring session*/
monitor-session mysession ethernet
    destination file
    tm-drop rx
!
```

Step 4 Verify the buffer drop mirroring configuration on the PWHE interface with the **show spp node-counters** command.

In the following example, you can verify the buffer drop mirroring configuration on the PWHE interface.

```
Router# show spp node-counters
Tue Feb 27 21:12:49.886 UTC
0/1/CPU0:
socket/rx
                ether raw pkts:              10
                low que accept:              10
           sent to XR classify:              10
-------------------------------
socket/tx
                       ce pkts:              20
        SW padding vector used:              20
-------------------------------
device/classify
      forwarded to spp clients:              10
 forwarded NPU packet to NetIO:              10
            L3 Route not found:              10
-------------------------------
client/inject
        pkts injected into spp:              10
```

```
  NetIO->CPU injected into spp:              10
  NetIO->CPU PKT IPV4_PREROUTE:              10
-------------------------------
pd_span_drop
                     SPAN drop:              10
-------------------------------
client/punt
              punted to client:              10
-------------------------------
```

Step 5 Perform these steps to capture and verify packets.

- Perform these steps to capture and verify buffer drop mirroring with the span to file feature:

a) Enable capturing the buffer drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

The router saves the pcap file in the temporary folder.

b) Stop capturing the buffer drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

c) Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

d) Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

e) Verify that the captured packet matches the original packet.

- Perform these steps to capture and verify buffer drop mirroring with ERSPAN feature:

a) Capture the packets with a traffic generator tool.

b) Verify that the captured packet is a GRE packet and should match the original sent packet encapsulated with a GRE header.

Preferred Tunnel Path

Table 21: Feature History Table

Feature Name	Release Information	Feature Description

VPLS over Preferred TE and MPLS OAM	Release 7.5.2	Based on your network traffic pattern, you can configure the preferred Traffic Engineering (TE) tunnel path between Provider Edge (PE) routers participating in the same Virtual Private LAN Services (VPLS). You optimize network resource utilization and performance when you set an explicit path on the PE router to direct traffic flow to a specific destination PE router. With VPLS, you now have MPLS-OAM capabilities for troubleshooting MPLS networks: • MPLS LSP Ping • MPLS LSP Traceroute • Flow-Aware Transport (FAT) Pseudowires (PW) This functionality adds the following command: control-word

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
```

Tunnel Configuration for VPLS

```
interface tunnel-te1
 ipv4 unnumbered Loopback0
 signalled-bandwidth 50
 destination 10.12.12.12
 path-option 1 explicit name FC1
```

L2 Configuration Example—VPLS over Preferred TE Tunnel

To configure VPLS over preferred TE tunnel, run the following commands:

```
RP/0/RP0/CPU0:r1(config)#interface FourHundredGigE0/0/0/0.1 l2transport
RP/0/RP0/CPU0:r1(config-subif)#encapsulation dot1q 100
RP/0/RP0/CPU0:r1(config-subif)#rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:r1(config-subif)#exit
RP/0/RP0/CPU0:r1(config)#l2vpn
RP/0/RP0/CPU0:r1(config)#pw-class c
RP/0/RP0/CPU0:r1(config-l2vpn-pwc)#encapsulation mpls
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#control-word
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#load-balancing
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls-load-bal)#flow-label both
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls-load-bal)#exit
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#preferred-path interface tunnel-te 1

RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#exit
RP/0/RP0/CPU0:r1(config-l2vpn-pwc)#exit
RP/0/RP0/CPU0:r1(config-l2vpn)#bridge group bg bridge-domain bd
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd)#interface FourHundredGigE0/0/0/0.1
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd)#neighbor 10.12.12.12 pw-id 100
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd-pw)#pw-class c
```

Verification

```
RP/0/RP0/CPU0:r1#show l2vpn bridge-domain detail
Wed Apr 20 17:53:26.232 UTC
Legend: pp = Partially Programmed.
Bridge group: bg, bridge-domain: bd, id: 0, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: Default
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 131072, Action: none, Notification: syslog
  MAC limit reached: no, threshold: 75%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  E-Tree: Root
  Dynamic ARP Inspection: disabled, Logging: disabled
  IP Source Guard: disabled, Logging: disabled
  DHCPv4 Snooping: disabled
  DHCPv4 Snooping profile: none
  IGMP Snooping: disabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 1
  Filter MAC addresses:
  P2MP PW: disabled
  Multicast Source: Not Set
  Create time: 20/04/2022 17:37:30 (00:15:55 ago)
  No status change since creation
  ACs: 1 (1 up), VFIs: 0, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: FourHundredGigE0/0/0/0, state is up
      Type Ethernet
```

```
     MTU 1500; XC ID 0x1; interworking none
     MAC learning: enabled
     Flooding:
       Broadcast & Multicast: enabled
       Unknown unicast: enabled
     MAC aging time: 300 s, Type: inactivity
     MAC limit: 131072, Action: none, Notification: syslog
     MAC limit reached: no, threshold: 75%
     MAC port down flush: enabled
     MAC Secure: disabled, Logging: disabled
     Split Horizon Group: none
     E-Tree: Root
     Dynamic ARP Inspection: disabled, Logging: disabled
     IP Source Guard: disabled, Logging: disabled
     DHCPv4 Snooping: disabled
     DHCPv4 Snooping profile: none
     IGMP Snooping: disabled
     IGMP Snooping profile: none
     MLD Snooping profile: none
     Storm Control: bridge-domain policer
     Static MAC addresses:
     Statistics:
       packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent
0
       bytes: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent 0
       MAC move: 0
     Storm control drop counters:
       packets: broadcast 0, multicast 0, unknown unicast 0
       bytes: broadcast 0, multicast 0, unknown unicast 0
     Dynamic ARP inspection drop counters:
       packets: 0, bytes: 0
     IP source guard drop counters:
       packets: 0, bytes: 0
     PD System Data: Learn key: 0
List of Access PWs:
   PW: neighbor 10.12.12.12, PW ID 100, state is up ( established )
     PW class c, XC ID 0xa0000001
     Encapsulation MPLS, protocol LDP
     Source address 10.10.10.10
     PW type Ethernet, control word enabled, interworking none
     PW backup disable delay 0 sec
     Sequencing not set
     Preferred path Active : tunnel-te1, Statically configured, fallback enabled
     Ignore MTU mismatch: Disabled
     Transmit MTU zero: Disabled
     Tunnel : Up

     PW Status TLV in use
       MPLS         Local                          Remote
       ------------ ------------------------------ ---------------------------
       Label        24000                          24000
       Group ID     0x0                            0x0
       Interface    Access PW                      Access PW
       MTU          1500                           1500
       Control word enabled                        enabled
       PW type      Ethernet                       Ethernet
       VCCV CV type 0x2                            0x2
                    (LSP ping verification)        (LSP ping verification)
       VCCV CC type 0x7                            0x7
                    (control word)                 (control word)
                    (router alert label)           (router alert label)
                    (TTL expiry)                   (TTL expiry)
       ------------ ------------------------------ ---------------------------
     Incoming Status (PW Status TLV):
```

```
  Status code: 0x0 (Up) in Notification message
MIB cpwVcIndex: 2684354561
Create time: 20/04/2022 17:37:30 (00:15:55 ago)
Last time status changed: 20/04/2022 17:53:22 (00:00:04 ago)
MAC withdraw messages: sent 0, received 0
Forward-class: 0
Static MAC addresses:
Statistics:
  packets: received 0 (unicast 0), sent 0
  bytes: received 0 (unicast 0), sent 0
  MAC move: 0
Storm control drop counters:
  packets: broadcast 0, multicast 0, unknown unicast 0
  bytes: broadcast 0, multicast 0, unknown unicast 0
MAC learning: enabled
Flooding:
  Broadcast & Multicast: enabled
  Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 131072, Action: none, Notification: syslog
MAC limit reached: no, threshold: 75%
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
E-Tree: Root
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: bridge-domain policer
```

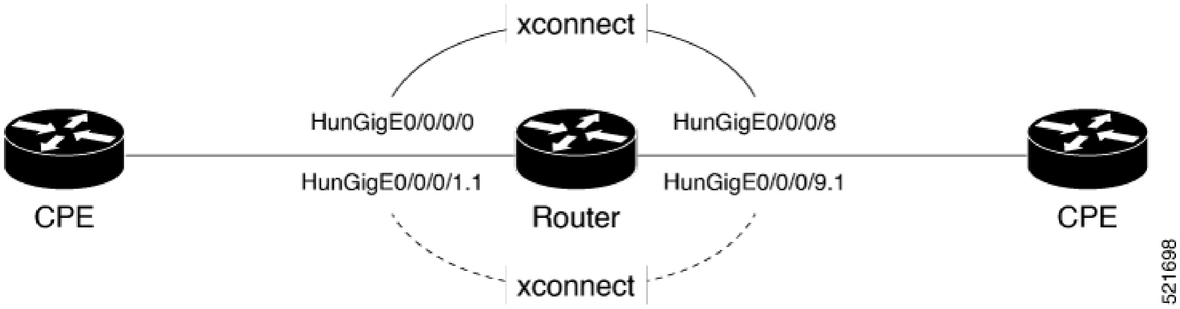
Configure Local Switching Between Attachment Circuits

Table 22: 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 20: 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

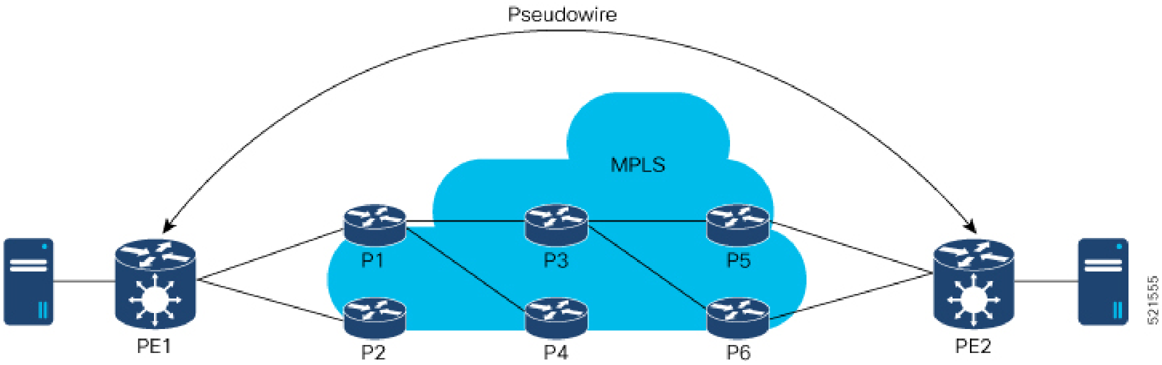
- Verify if the configured cross-connect is UP

```
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 23: 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.

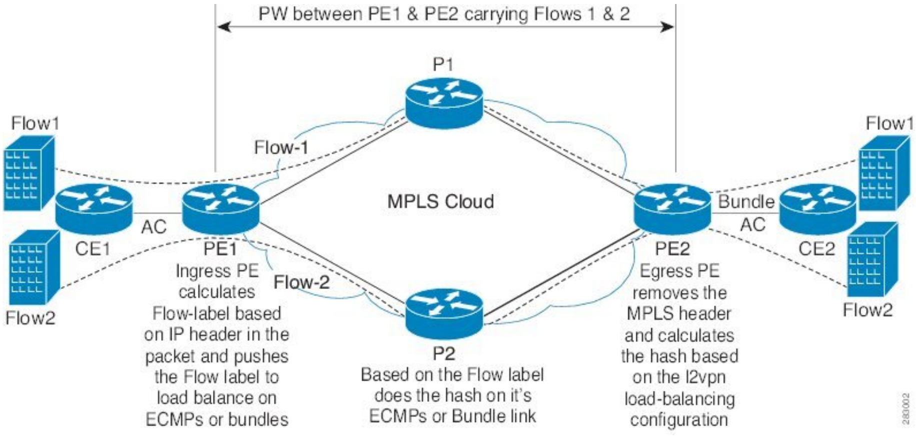
Load balancing using Flow-Label

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.


PW between PE1 & PE2 carrying Flows 1 & 2
P1
Flow1
Flow-1
MPLS Cloud
Flow1
Bundle
AC
AC
CE1
PE1
Ingress PE calculates Flow-label based on IP header in the packet and pushes the Flow label to load balance on ECMPs or bundles
Flow-2
Flow2
P2
Based on the Flow label does the hash on it's ECMPs or Bundle link
PE2
Egress PE removes the MPLS header and calculates the hash based on the l2vpn load-balancing configuration
CE2
Flow2
283002

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

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Introduced New VLAN Tag Format Support for Load Balancing	Release 24.3.1	Introduced in this release on: Fixed Systems (8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Introduced support for the following VLAN tags on line cards and routers with Q100, Q200, and P100 based Silicon One ASIC: • Single VLAN tag 0x88A8 • QinQ with outer 0x8100 and inner 0x8100 • QinQ with outer 0x9100 and inner 0x8100 Introduced support for BUM traffic in VPLS service load balancing.
Extended L2VPN Traffic Load Balancing Support for line cards and routers with the P100 based Silicon One ASIC	Release 24.1.1	Introduced support for the following VLAN tags on line cards and routers with P100 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ outer 0x88A8 and inner 0x8100 Introduced support for QinQ with outer 0x8100 and inner 0x8100 on line cards and routers with Q200 based Silicon One ASIC.
Extended L2VPN Traffic Load Balancing Support for line cards and routers with the Q200 based Silicon One ASIC	Release 7.5.1	Introduced support for the following VLAN tags on line cards and routers with Q200 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ outer 0x88A8 and inner 0x8100

Feature Name	Release Information	Feature Description
L2VPN Traffic Load Balancing on PE Router	Release 3.7.1	Distributes L2 VPN traffic across multiple physical links or paths. Introduced support for the following VLAN tags on line cards and routers with Q100 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ with outer 0x88A8 and inner 0x8100

A Provider Edge (PE) router balances Layer 2 Virtual Private Network (L2VPN) traffic by efficiently distributing it across network paths using various load balancing techniques.

On Cisco 8000, different load balance methods are used for different traffic groups:

1. Traffic in VPWS (E-Line) Service - For more information, see VPWS Service and Unicast Traffic in VPLS Service Load Balancing, on page 116
2. Unicast Traffic in VPLS (E-LAN) Service - For more information, see VPWS Service and Unicast Traffic in VPLS Service Load Balancing, on page 116.
3. Broadcast, Unknown Unicast, and Multicast (BUM) Traffic in VPLS (E-LAN) Service - For more information, see BUM Traffic in VPLS Service Load Balancing, on page 116.

Load balance is orthorgnal to point-to-point or multi-point connectivity. Load balance is needed over

1. Equal-Cost Multi-Path (ECMP) paths
2. Link Aggregation (LAG) bundle members

Supported VLAN Tag Formats for Load Balancing

The Cisco 8000 load balances traffic when it is either not VLAN tagged or tagged with VLAN in supported formats.

Table 25: Supported VLAN Tag Formats

VLAN Format	Q100 supports VLAN from Release	Q200 supports VLAN from Release	P100 supports VLAN from Release	A100 supports VLAN from Release
Single VLAN Tag 0x8100	3.7.1	7.5.1	24.1.1	24.1.1
Single VLAN Tag 0x88A8	24.3.1	24.3.1	24.3.1	24.3.1
QinQ outer 0x88A8, inner 0x8100 (double VLAN tags)	3.7.1	7.5.1	24.1.1	24.1.1

VLAN Format	Q100 supports VLAN from Release	Q200 supports VLAN from Release	P100 supports VLAN from Release	A100 supports VLAN from Release
QinQ outer 0x8100, inner 0x8100 (double VLAN tags)	24.3.1	24.1.1	24.3.1	24.3.1
QinQ outer 0x9100, inner 0x8100 (double VLAN tags)	24.3.1	24.3.1	24.3.1	24.3.1

VPWS Service and Unicast Traffic in VPLS Service Load Balancing

The router performs load balancing on outgoing interfaces and bundle members in these scenarios:

1. Ethernet frames entering from a L2 main or sub interface may be
   - switched out to another L2 interface that is part of a bundle, or
   - routed out to L3 interfaces with MPLS encapsulation.
2. MPLS labeled PW and EVPN traffic entering from an L3 interface. After label disposition, the customer Ethernet frames may be
   - switched out to an L2 main or sub interface that is part of a bundle, or
   - routed out to L3 interfaces with MPLS encapsulation.

The router parses packets to identify the required headers for generating a load balance hash, which determines the path to route the traffic across the network. The hashing process varies based on whether the traffic is received on L2 or L3 interfaces.

For load balance hash on traffic received from L2 interfaces, refer to Hash for Traffic Received on L2 Interface, on page 117.

For load balance hash on traffic received from L3 interfaces, refer to Hash for Traffic Received on L3 Interface, on page 118.

BUM Traffic in VPLS Service Load Balancing

The router performs BUM Traffic load balancing on outgoing interfaces and bundle members in these scenarios:

- Sending Traffic to an L2 Interface on bundle
- Sending Traffic over an MPLS PW
- Sending Traffic to EVPN MPLS Network

Sending Traffic to an L2 Interface on Bundle

When sending traffic to a L2 interface, the router does not perform load balancing over bundle members. Instead, it pins all traffic sent to an L2 main or sub-interface to a single bundle member. The selection of the bundle member is based on the main or sub-interface ID.

Sending Traffic over an MPLS PW

When BUM traffic is routed to an MPLS PW, the traffic is routed out to L3 interfaces. The router performs ECMP and bundle load balancing on the PW traffic. The hashing for load balancing is based on:

- PW VC label and flow label
- The 12 bytes of the PW payload. If control word (CW) is enabled, this includes a combination of 4 bytes of CW and 8 bytes of inner Ethernet MAC address.

Sending Traffic to EVPN MPLS Network

When BUM traffic is routed to an EVPN MPLS network, the traffic is encapsulated with:

- EVPN label
- ETREE leaf label or Ethernet Segment split horizon label

The MPLS encapsulated traffic is routed out to L3 interfaces. ECMP and bundle load balancing are performed on the EVPN MPLS encapsulated traffic. The hashing for load balancing is based on:

- EVPN label
- ETREE leaf label or Ethernet Segment split horizon label
- The 12 bytes of the PW payload. If CW is enabled, this includes a combination of 4 bytes of CW and 8 bytes of inner Ethernet MAC address.

MPLS non-IP Hash Disabled Configuration

In certain configurations, you may choose to disable the non-IP hash mode. When non-IP hash mode is disabled, the Ethernet MAC address of BUM traffic isn’t used to perform load balance hashing. This can impact how BUM traffic is distributed across the network, as the router relies on other available headers for hashing.

Hash for Traffic Received on L2 Interface

For traffic received on a L2 interface, hashing depends on the headers present in the packet stack. The router generates the load balance hash using the designated fields based on the packet stack headers.

When ethernet frames entering a L2 main or sub interface, the router identifies the IP header within the packet based on the packet stack headers. The router uses these packet stack headers to generate the hash for traffic received on the L2 interface.

IP Traffic on L2 Interface

An ethernet frame is classified as IP traffic if it meets the following requirements:

- The ethernet header contains no more than two VLAN tags.
- The ethernet header either has no VLAN tag or has VLAN tags in a supported format as listed in Supported VLAN Tag Formats for Load Balancing, on page 115.
- No more than ten MPLS labels are placed in front of the IP header.

Non-IP Traffic on L2 Interface

All traffic that does not meet the description of IP traffic on L2 Interface is classified as non-IP traffic on L2 interface.

L2 Hash Fields

The L2 hash fields include the source and destination MAC addresses and the outer VLAN ID.

L3 Hash Fields

The L3 hash fields include the source and destination IP addresses and the source and destination ports of the layer 4 header.

IP Traffic Load Balancing

1. On Q100 and Q200 based Silicon One ASIC:

   Before Release 24.2.1, L2 hash fields were used in hashing. L3 hash fields were also included for limited traffic types.

   Starting from the Release 24.2.1, both L2 and L3 fields are used in hashing for all IP traffic.

2. On P100 and A100 based Silicon One ASIC:

   L2 hash fields are used in hashing. L3 hash fields are also included for limited traffic types.

Non-IP Traffic Load Balancing

Non-IP traffic on the L2 interface is load balanced using L2 hash fields.

If any of the designated fields are missing, the router replaces those field values with zeros.

Hash for Traffic Received on L3 Interface

For traffic received on a L3 interface hashing depends on several criteria, including the headers present in the packet stack and whether the Control Word (CW) and Flow Label (FL) are enabled on the pseudowire (PW).

When ethernet frames entering a L3 interface, the router identifies the IP header within the packet based on the packet stack headers.

IP over PW Traffic

An ethernet frame is classified as IP over PW traffic if it meets the following criteria:

- The frame contains an outer ethernet header and an inner ethernet header.
- No more than two MPLS labels are placed between the outer ethernet header and the inner ethernet header.
- There is no control word in front of the inner ethernet header.
- The inner ethernet header contains no more than two VLAN tags.
- The inner ethernet header has no VLAN tag or has VLAN tags in supported format as listed in Supported VLAN Tag Formats for Load Balancing, on page 115.
- Behind the inner ethernet header, there are no more than ten MPLS labels placed in front of the IP header.

Non-IP over PW Traffic

All traffic that does not meet the IP over PW traffic criteria is classified as non-IP over PW traffic.

Inner Ethernet L2 Hash Fields

The inner ethernet L2 hash fields include the source and destination MAC addresses, and the first VLAN tag of the inner ethernet frame.

Inner Ethernet L3 Hash Fields

The inner ethernet L3 hash fields include L3 and L4 headers in the inner ethernet frame. They are source and destination IP addresses, the source and destination ports of the layer 4 header.

Control Word Presence L2 Hash Fields

The 12 bytes of the PW payload, which include 4 bytes of CW followed by the 8 bytes of inner ethernet frame MAC address.

MPLS Hash Fields of Outer Ethernet Frame

All MPLS labels in front of inner ethernet header.

PW Disposition Traffic Load Balancing

1. PW disposition traffic load balance when control word and flow label are both disabled.

   IP over PW Traffic Load Balancing

   a. On Q100 and Q200 based Silicon One ASIC:

      IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

      Before Release 24.2.1, inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

      Starting from Release 24.2.1, both inner ethernet L2 hash fields and inner ethernet L3 hash fields are used in hashing.

   b. On P100 and A100 based Silicon One ASIC:

      IP over PW traffic load balance hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

   Non-IP over PW Traffic Load Balancing

   Non-IP over PW traffic load balance hash always uses the inner ethernet L2 hash fields.

2. PW disposition traffic load balance when control word is enabled and flow label disabled.

   In this case, all PW traffic is non-IP over PW traffic. Load balance hash uses control word presence L2 hash fields.

3. PW disposition traffic load balance when control word is disabled and flow label enabled.

   a. On Q100 and Q200 based Silicon One ASIC:

      IP over PW Traffic Load Balancing

      IP over PW traffic load balancing hash uses the following fields:

      - Before Release 24.2.1, hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.
      - In Release 24.2.1, inner ethernet L2 hash fields and inner ethernet L3 hash fields are both used in hashing.

- Starting from Release 24.3.1, inner ethernet L3 hash fields and MPLS hash fields of outer ethernet frame are used in hashing.

Non-IP over PW Traffic Load Balancing

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Starting from Release 24.3.1 release, MPLS hash fields of outer ethernet frame are used in hashing.

b. On P100 and A100 based Silicon One ASIC:

IP over PW Traffic Load Balancing

IP over PW traffic load balance hash uses the following fields:

- Before Release 24.3.1, hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.
- Starting from Release 24.3.1, hash uses MPLS hash fields of outer ethernet frame. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

Non-IP over PW Traffic Load Balancing

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Starting from Release 24.3.1 release, MPLS hash fields of outer ethernet frame are used in hashing.

4. PW disposition traffic load balance when control word and flow label are both enabled.

   In this case, all PW traffic is Non-IP over PW traffic.

   Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the control word presence L2 hash fields.

   Starting from Release 24.3.1, MPLS hash fields of outer ethernet frame are used in hashing.

G.8032 Ethernet Ring Protection Switching

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
G.8032 Ethernet Ring Protection Switching	Release 24.2.11	Ethernet Ring Protection Switching (ERPS) protocol, defined in ITU-T G.8032, provides protection for Ethernet traffic in a ring topology, while ensuring that there are no loops within the ring at the Ethernet layer. The loops are prevented by blocking either a predetermined link or a failed link. This feature introduces the **ethernet ring g8032** and **ethernet ring g8032 profile** commands.

ERPS ensures that link or node failures recover faster in Ethernet ring topologies. During a link failure, it reroutes traffic to provide continuous connectivity, simplifies network management, and operates independently of the control planes.

Overview

Each Ethernet ring node is connected to adjacent Ethernet ring nodes participating in the Ethernet ring using two independent links. A ring link never allows the formation of loops that affect the network. The Ethernet ring uses a specific link to protect the entire Ethernet ring. This specific link is called the ring protection link (RPL). A ring link is bound by two adjacent Ethernet ring nodes and a port for a ring link (also known as a ring port).

Note The minimum number of Ethernet ring nodes in an Ethernet ring is two.

The fundamentals of ring protection switching are:

- The principle of loop avoidance
- The utilization of learning, forwarding, and Filtering Database (FDB) mechanisms

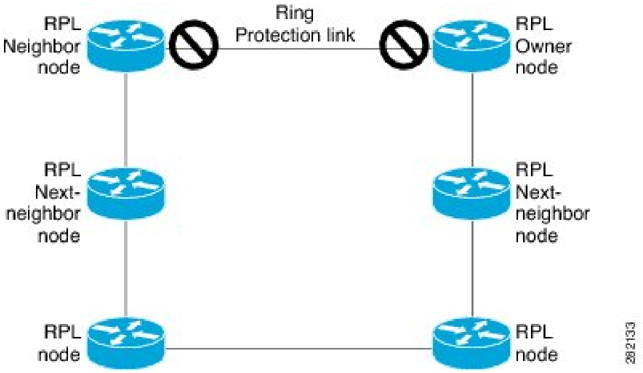
Loop avoidance in an Ethernet ring is achieved by ensuring that, at any time, traffic flows on all but one of the ring links which is the RPL. Multiple nodes are used to form a ring:

- RPL owner - It’s responsible for blocking traffic over the RPL so that no loops are formed in the Ethernet traffic. There can be only one RPL owner in a ring.
- RPL neighbor node - The RPL neighbor node is an Ethernet ring node next to the RPL. It's responsible for blocking its end of the RPL under normal conditions. This node type is optional and prevents RPL usage when protected.

- RPL next-neighbor node - The RPL next-neighbor node is an Ethernet ring node next to the RPL owner node or RPL neighbor node. It's used for FDB flush optimization on the ring. This node is also optional.

The following figure illustrates the G.8032 Ethernet ring.

Figure 21: G.8032 Ethernet Ring



Nodes on the ring use control messages called RAPS to coordinate the activities of switching on or off the RPL link. Any failure along the ring triggers a RAPS signal fail (RAPS SF) message along both directions, from the nodes next to the failed link, after the nodes have blocked the port facing the failed link. On obtaining this message, the RPL owner unblocks the RPL port.

Note A single link failure in the ring ensures a loop-free topology.

Line status and Connectivity Fault Management protocols are used to detect ring link and node failure. During the recovery phase, when the failed link is restored, the nodes next to the restored link send RAPS no request (RAPS NR) messages. On obtaining this message, the RPL owner blocks the RPL port and sends RAPS no request, root blocked (RAPS NR, RB) messages. This causes all other nodes, other than the RPL owner in the ring, to unblock all blocked ports. The ERPS protocol is robust enough to work for both unidirectional failure and multiple link failure scenarios in a ring topology.

A G.8032 ring supports these basic operator administrative commands:

- Force switch (FS) - Allows the operator to forcefully block a particular ring-port.
  - Effective even if there’s an existing SF condition.
  - Multiple FS commands for ring supported.
  - May be used to allow immediate maintenance operations.
- Manual switch (MS) - Allows the operator to manually block a particular ring-port.
  - Ineffective in an existing FS or SF condition.
  - Overridden by new FS or SF conditions.
  - Multiple MS commands cancel all MS commands.
- Clear - Cancels an existing FS or MS command on the ring-port.

    - Used (at RPL Owner) to clear non-revertive mode.

A G.8032 ring can support multiple instances. An instance is a logical ring running over a physical ring. Such instances are used for various reasons, such as load balancing VLANs over a ring. For example, odd VLANs may go in one direction of the ring, and even VLANs may go in the other direction. Specific VLANs can be configured under only one instance. They cannot overlap multiple instances. Otherwise, data traffic or RAPS packets can cross logical rings, and that isn't desirable.

G.8032 ERPS provides a new technology that relies on line status and Connectivity Fault Management (CFM) to detect link failure. By running CFM Continuity Check Messages (CCM) messages at an interval of 100ms, it's possible to achieve SONET-like switching time performance and loop free traffic.

For more information about Ethernet Connectivity Fault Management (CFM) and Ethernet Fault Detection (EFD) configuration, refer to the *Configuring Ethernet OAM on the Cisco 8000 Series Router* module in the *Cisco 8000 Series Router Component Configuration Guide*.

Timers

G.8032 ERPS specifies the use of different timers to avoid race conditions and unnecessary switching operations:

- Delay Timers - used by the RPL Owner to verify that the network has stabilized before blocking the RPL.
    - After SF condition, a Wait-to-Restore (WTR) timer is used to verify that SF isn’t intermittent. The WTR timer can be configured by the operator, and the default time interval is 5 minutes. The time interval ranges 1–12 minutes.
    - After the FS/MS command, a Wait-to-Block timer is used to verify that no background condition exists.

Note The Wait-to-Block timer may be shorter than the Wait-to-Restore timer.

- Guard Timer - used by all nodes when changing state; it blocks latent outdated messages from causing unnecessary state changes. The Guard timer can be configured and the default time interval is 500 ms. The time interval ranges 10-2000 ms.
- Hold-off timers - used by the underlying Ethernet layer to filter out intermittent link faults. The hold-off timer can be configured and the default time interval is 0 seconds. The time interval ranges 0–10 seconds.
    - Faults are reported to the ring protection mechanism, only if this timer expires.

During a link failure, the G8032 EPR performs either of the following operations to provide continuous connectivity:

- If it’s unable to recover from the link failure, it reroutes traffic. For more information, refer to Protection Switching during Single Link Failure, on page 124.
- Wait to recover from link failure to prevent unnecessary switching operations. For more information, refer to Recovery from Single Link Failure, on page 125.

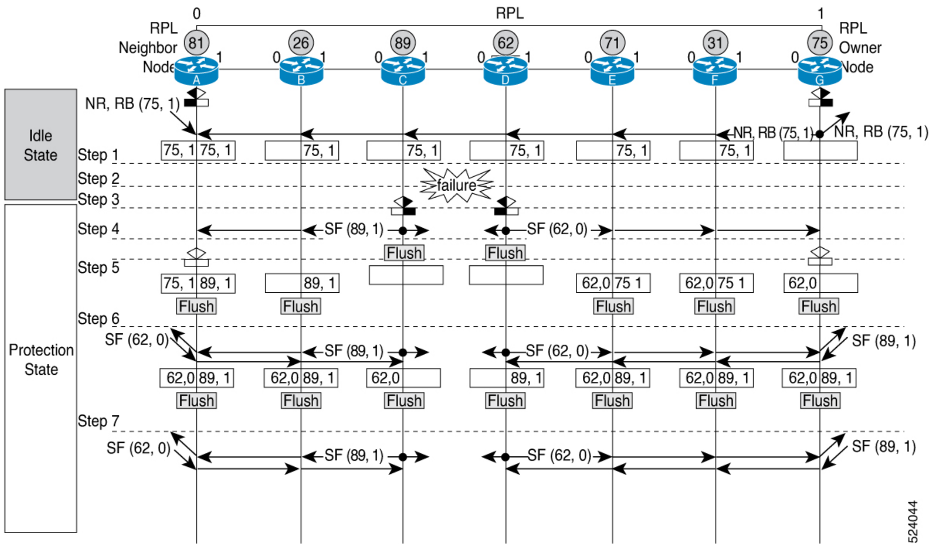
Protection Switching during Single Link Failure

The following example describes the protection switching process during a single link failure:

For example, consider the Figure 22: Protection Switching during G.8032 Single Link Failure, on page 124 figure with the following configuration:

- An ethernet ring is composed of seven ethernet ring nodes (A to G) with node ID (81, 26, 89, 62, 71, 31, and 75) and port ID (0 and 1).
- The ethernet ring node A is the RPL neighbor node.
- The ethernet ring node G is the RPL owner node.
- The RPL is the ring link between ethernet ring nodes A and G.
- Traffic is blocked at both ends of the RPL.

Figure 22: Protection Switching during G.8032 Single Link Failure



The ERPS performs the following protection switching steps during a single link failure:

1. The link operates in the normal condition.
2. A failure occurs between ring nodes C and D.
3. Ethernet ring nodes C and D detect a local signal failure (SF) condition and after the holdoff time interval, block the failed ring port and perform the forwarding database (FDB) flush.
4. Ethernet ring nodes C and D start sending ring automatic protection switching (RAPS) (SF) messages periodically along with the node ID and Blocked Port Ring (BPR) pair on both ring ports, while the SF condition persists.

   For example, ring node C sends the SF(89,1) message, which consists of node ID 89 and BPR 1.
5. All Ethernet ring nodes receiving an RAPS (SF) message perform FDB flush. When the RPL owner node G and RPL neighbor node A receive an RAPS (SF) message, the Ethernet ring node unblocks its end of the RPL and performs the FDB flush.

6. All Ethernet ring nodes receiving a second RAPS (SF) message perform the FDB flush again; this is because of the Node ID and BPR-based mechanism.
7. Stable SF condition—RAPS (SF) messages on the Ethernet Ring. Further RAPS (SF) messages trigger no further action.

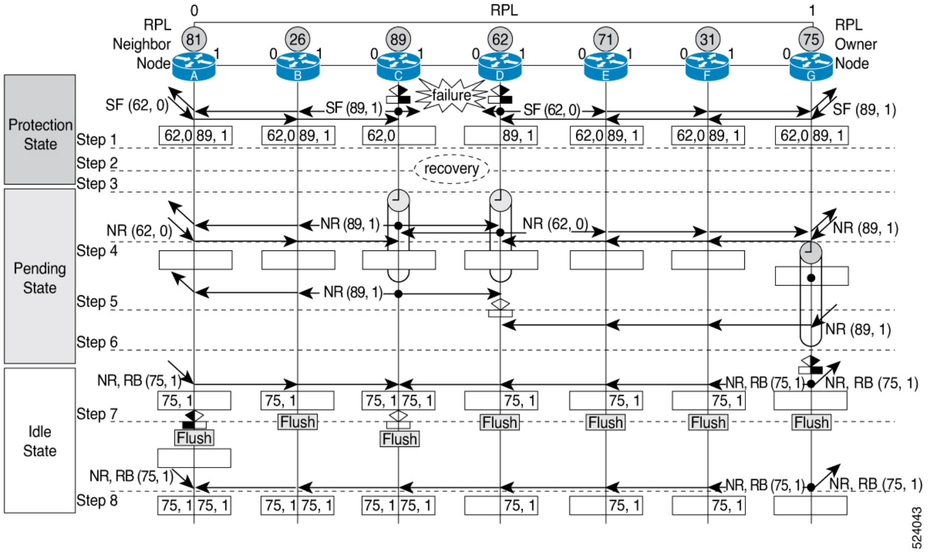
Recovery from Single Link Failure

The following example describes the single link failure recovery process:

For example, consider the Figure 23: Single link failure Recovery (Revertive operation), on page 125 figure with the following configuration:

- An ethernet ring is composed of seven ethernet ring nodes (A to G) with node ID (81, 26, 89, 62, 71, 31, and 75) and port ID (0 and 1).
- The ethernet ring node A is the RPL neighbor node.
- The ethernet ring node G is the RPL owner node.
- The RPL is the ring link between ethernet ring nodes A and G.
- Traffic is blocked at both ends of the RPL.

Figure 23: Single link failure Recovery (Revertive operation)



The ERPS performs the following reversion steps to revertive the link during a single link failure:

1. A failure occurs between ring nodes C and D.
2. Recovery of link failure occurs between ring nodes C and D.
3. Ethernet ring nodes C and D detect clearing of SF condition, start the guard timer and initiate periodical transmission of RAPS No Request (NR) messages on both ring ports.

Note The guard timer prevents the reception of RAPS messages.

4. When the Ethernet ring nodes receive an RAPS (NR) message, the node ID and BPR pair of a receiving ring port is deleted and the RPL owner node starts the WTR timer.
5. When the guard timer expires on ethernet ring nodes C and D, they may accept the new RAPS messages that they receive. Ethernet ring node D receives an RAPS (NR) message with higher Node ID from ethernet ring node C, and unblocks its non-failed ring port.
6. When the WTR timer expires, the RPL owner node blocks its end of the RPL, sends the RAPS (NR, RB) message with the node ID and BPR pair, and performs the FDB flush.
7. When Ethernet ring node C receives an RAPS (NR, RB) message, it removes the block on its blocked ring ports, and stops sending RAPS (NR) messages. On the other hand, when the RPL neighbor node A receives an RAPS (NR, RB) message, it blocks its end of the RPL. In addition to this, Ethernet ring nodes A to F perform the FDB flush when receiving an RAPS (NR, RB) message, due to the existence of the Node ID and BPR based mechanism.
8. Link operates in the stable SF condition.

Restrictions for G.8032 Ethernet Ring Protection Switching

- You must not configure G.8032 ERPS and CFM down-mep on the same sub-interface. If you enable it, then the router displays a syslog message, as shown in the following example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# ethernet ring g8032 test
Router(config-l2vpn-erp)# port0 interface FourHundredGigE0/0/0/6
Router(config-l2vpn-erp)# port1 interface FourHundredGigE0/0/0/10
Router(config-l2vpn-erp)# instance 1
Router(config-l2vpn-erp-inst)# profile test
Router(config-l2vpn-erp-inst)# rpl port0 owner
Router(config-l2vpn-erp-inst)# inclusion-list vlan-ids 1,100
Router(config-l2vpn-erp-inst)# aps-channel
Router(config-l2vpn-erp-inst-aps)# port0 interface FourHundredGigE0/0/0/6.1
Router(config-l2vpn-erp-inst-aps)# port1 interface FourHundredGigE0/0/0/10.1

Router(config)# interface FourHundredGigE0/0/0/6.1 l2transport
Router(config-if)# encapsulation dot1q 1
Router(config-if)# ethernet cfm
Router(config-if-cfm)# mep domain domain1 service link1 mep-id 2

%PLATFORM-SPITFIRE_CFM-3-G8032_VIOLATION : G8032 has been configured for interface
FourHundredGigE0/0/0/6.1
where CFM configuration exists. G8032 config is disabled.
```

  Instead configure the CFM down-mep on the main interface and configure the G.8032 ERPS on the sub-interface.

- Linecards and fixed routers with Q100 and Q200 based Silicon One ASICs don't support G.8032 Ethernet Ring Protection Switching.

Configuring G.8032 Ethernet Ring Protection Switching

To configure the G.8032 operation, you have to configure ERPS and CFM separately as follows:

- Configure the ERPS profile, ERPS instance, ERPS paramenters, and TCN propagation by including the following requirements:
  - Designate a (sub)interface which is used as the APS channel.
  - Designate a (sub)interface which is monitored by CFM.
  - Verify whether the interface is an RPL link, and, if it is a RPL link then indicate the RPL node type.

  For more information, see the following sections:
  - Configuring ERPS Profile, on page 127
  - Configuring an ERPS Instance, on page 128
  - Configuring ERPS Parameters, on page 130
  - Configuring TCN Propagation, on page 131
- Configure CFM with EFD to monitor the ring links. For more information, see the Configuring CFM MEP, on page 131.

Note MEP for each monitor link needs to be configured with different Maintenance Association.

- The bridge domains to create the Layer 2 topology. The RAPS channel is configured in a dedicated management bridge domain separated from the data bridge domains.
- Behavior characteristics, that apply to ERPS instance, if different from default values. This is optional.

This section provides information on:

Configuring ERPS Profile

Perform this task to configure the Ethernet Ring Protection Switching (ERPS) profile.

Step 1 Configure a new G.8032 ERPS profile using the **Ethernet ring g8032 profile** command.

Example:

```
Router# configure
Router(config)# Ethernet ring g8032 profile p1
```

Enables G.8032 ring mode, and enters G.8032 configuration submode.

Step 2 Sets the hold-off timer using the **timer** command.

Example:

```
Router(config-g8032-ring-profile)# timer hold-off 5
```

Specifies a time interval (in seconds) for the guard, hold-off, and wait-to-restore timers.

The hold-off timer prevents unnecessary switching due to short-lived failures on the ring.

Step 3 Specify a non-revertive ring instance using the **non-revertive** command.

Example:

```
Router(config-g8032-ring-profile)# non-revertive
Router(config-g8032-ring-profile)# commit
```

This feature enables the router to use the current path until an administrator manually reverts to the original path.

Configuring an ERPS Instance

Perform this task to configure an ERPS instance.

Step 1 Configure the layer 2 VPN with a bridge group.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group cisco
Router(config-l2vpn-bg)# bridge-domain bd1
```

Creates a bridge group that can contain bridge domains, and then assigns network interfaces to the bridge domain.

Step 2 Configure a bridge domain for R-APS channels using the **bridge-domain** command.

Example:

```
Router(config-l2vpn-bg)# bridge-domain bd1
```

Establishes a bridge domain for R-APS channels, and enters L2VPN bridge group bridge domain configuration mode.

Step 3 Configure an interface to a bridge domain on ports 0 and 1 using the **interface** command.

Example:

```
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 4 Configure a bridge domain for data traffic using the **bridge-domain** command.

Example:

```
Router(config-l2vpn-bg)# bridge-domain bd2
```

Establishes a bridge domain for data traffic, and enters L2VPN bridge group bridge domain configuration mode.

Step 5 Configure an interface to a bridge domain using the **interface** command.

Example:

```
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.10
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 6 Configure an ethernet ring using the **ethernet ring g8032** command.

Example:

```
Router(config-l2vpn)# ethernet ring g8032 r1
```

Enables G.8032 ring mode, and enters G.8032 configuration submode.

Step 7 Configure an ERPS instance using the **instance** command.

Example:

```
Router(config-l2vpn-erp)# instance 1
```

Enters the Ethernet ring G.8032 instance configuration submode.

Step 8 Add a description for the ERPS instance that you are configuring using the **description** command.

Example:

```
Router(config-l2vpn-erp-instance)# description test
```

Specifies a string that serves as a description for that instance.

Step 9 Configure an ERPS profile using the **profile** command.

Example:

```
Router(config-l2vpn-erp-instance)# profile p1
```

Specifies associated Ethernet ring G.8032 profile.

Step 10 Specify the RPL port and designates it as a neighbor using the **rpl** command.

Example:

```
Router(config-l2vpn-erp-instance)# rpl port0 neighbor
```

Specifies one ring port on the local node as RPL owner, neighbor, or next-neighbor.

Step 11 Configure the VLANs that are included in the ERPS instance using the **inclusion-list vlan-ids** command.

Example:

```
Router(config-l2vpn-erp-instance)# inclusion-list vlan-ids e-g
```

Associates a set of VLAN IDs with the current instance.

Step 12 Enable Automatic Protection Switching (APS) channel configuration mode for the ERPS instance and sets the priority level for the APS protocol.

Example:

```
Router(config-l2vpn-erp-instance)# aps-channel
Router(config-l2vpn-erp-instance-aps)# level 5
```

Enters the ethernet ring G.8032 instance aps-channel configuration submode and specifies the APS message level. The range is 0–7.

Step 13 Assign a port to the G.8032 APS channel interface.

Example:

```
Router(configl2vpn-erp-instance-aps)# port0 interface GigabitEthernet 0/0/0/0.1
```

Associates G.8032 APS channel interface to port0.

Step 14 Assign a port to the G.8032 APS channel interface.

Example:

```
Router(config-l2vpn-erp-instance-aps)# port1 interface GigabitEthernet 0/0/0/1.1
```

Associates G.8032 APS channel interface to port1.

Configuring ERPS Parameters

Perform this task to configure ERPS parameters.

Step 1 Configure an ethernet ring in L2VPN configuration mode.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# ethernet ring g8032 r1
```

Enters L2VPN configuration mode, enables G.8032 ring mode, and enters G.8032 configuration submode.

Step 2 Enable G.8032 ERPS for the specified port (ring port).

Example:

```
Router(config-l2vpn-erp)# port0 interface GigabitEthernet 0/0/0/0
```

Step 3 Specify a port to monitor the G.8032 ERPS and detect ring link failure.

Example:

```
Router(config-l2vpn-erp-port0)# monitor port0 interface 0/0/0/0.5
```

Specifies the port that is monitored to detect ring link failure per ring port. The monitored interface must be a sub-interface of the main interface.

Step 4 Exit from port configuration submode.

Example:

```
Router(config-l2vpn-erp-port0)# exit
```

Exits port0 configuration submode.

Step 5 Enable G.8032 ERPS for the specified port (ring port).

Example:

```
Router(config-l2vpn-erp)# port1 interface GigabitEthernet 0/0/0/1
```

Enables G.8032 ERPS for the specified port (ring port).

Step 6 Specify a port to monitor the G.8032 ERPS and detect ring link failure.

Example:

```
Router(config-l2vpn-erp-port1)# monitor port1 interface 0/0/0/1.5
```

Specifies the port that is monitored to detect ring link failure per ring port. The monitored interface must be a sub-interface of the main interface.

Step 7 Exit from port configuration submode.

Example:

```
Router(config-l2vpn-erp-port1)# exit
```

Exits port1 configuration submode.

Step 8 Configure a set of VLAN IDs that isn't protected by the Ethernet ring protection mechanism using the **exclusion-list vlan-ids** command.

Example:

```
Router(config-l2vpn-erp)# exclusion-list vlan-ids a-d
```

Step 9 Configure the ethernet ring G.8032 as an open ring.

Example:

```
Router(config-l2vpn-erp)# open-ring
```

Configuring TCN Propagation

Perform this task to configure topology change notification (TCN) propagation.

Enable TCN propagation in L2VPN configuration mode.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# tcn-propagation
Router(config-l2vpn)# commit
```

Enters L2VPN configuration mode and allows TCN propagation from minor ring to major ring and from MSTP to G.8032.

Configuring CFM MEP

Configuring the CFM on the main interface and G.8032 ERPS on the sub-interfaces allows you to constantly monitor the Ethernet ring's status. If a link in the ring fails, the Ethernet Fault Detection (EFD) shuts down the affected port and reroutes the traffic through the new path.

Perform the following steps to configure the CFM MEP:

Step 1 Configure a CFM domain and service.

Example:

```
Router# configure
Router(config)# ethernet cfm
Router(config-cfm)# domain dom23to24 level 6
Router(config-cfm-domain)# service ser23to24 down-meps
```

Step 2 Configure the continuity checks.

Example:

```
Router(config-cfm-svc)# continuity-check interval 10s
```

Step 3 Configure a MEP crosscheck on the main interface to detect failures and reroute the traffic.

Example:

```
Router(config-cfm-svc)# mep crosscheck
Router(config-cfm-svc-xcheck)# mep-id 3
Router(config-cfm-svc-xcheck)# exit
```

Step 4 Configure Ethernet Fault Detection (EFD) to detect failures and reroute the traffic.

Example:

```
Router(config-cfm-svc)# efd
```

Step 5 Configure CFM MEP on the sub-interface.

Example:

```
Router# configure terminal
Router(config)# interface Gigabiteethernet0/0/0/0.5
Router(config-if)# ethernet cfm
Router(config-if-cfm)# mep domain dom23to24 service ser23to24 mep-id 4
```

For more information about Ethernet Connectivity Fault Management (CFM), refer to the *Configuring Ethernet OAM on the Cisco 8000 Series Router* module in the *Cisco 8000 Series Router Interface and Hardware Component Configuration Guide*.

Configuring G.8032 Ethernet Ring Protection Switching: Example

This sample configuration illustrates the elements that a complete G.8032 configuration includes:

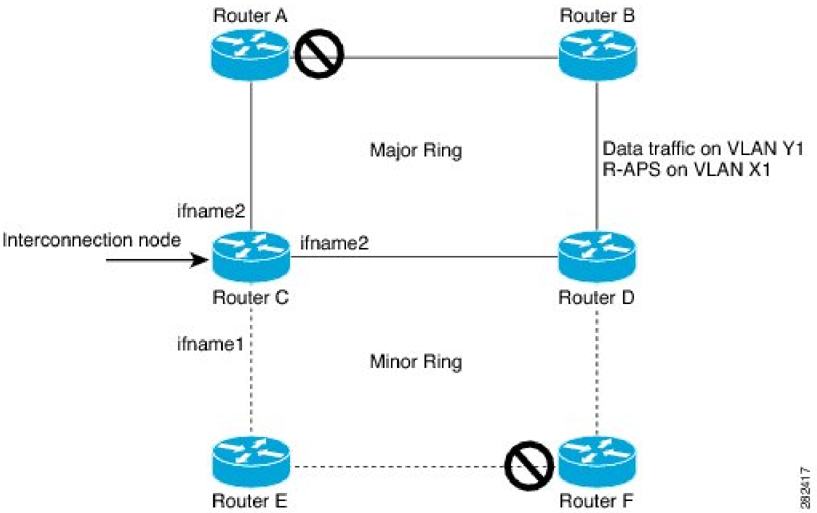
```
# Configure the ERP profile characteristics if ERPS instance behaviors are non-default.
ethernet ring g8032 profile ERP-profile
  timer wtr 60
  timer guard 100
  timer hold-off 1
  non-revertive


# Configure the ERPS instance under L2VPN
l2vpn
  ethernet ring g8032 RingA
    port0 interface g0/0/0/0
    port1 interface g0/1/0/0
    instance 1
      description BD2-ring
      profile ERP-profile
      rpl port0 owner
      vlan-ids 10-100
      aps channel
        level 3
        port0 interface g0/0/0/0.1
        port1 interface g1/1/0/0.1

# Set up the bridge domains
bridge group ABC
    bridge-domain BD2
      interface Gig 0/0/0/0.2
      interface Gig 0/1/0/0.2
      interface Gig 0/2/0/0.2

    bridge-domain BD2-APS
      interface Gig 0/0/0/0.1
      interface Gig 1/1/0/0.1
```

```
# EFPs configuration
interface Gig 0/0/0/0.1 l2transport
  encapsulation dot1q 5

interface Gig 1/1/0/0.1 l2transport
  encapsulation dot1q 5

interface g 0/0/0/0.2 l2transport
  encapsulation dot1q 10-100

interface g 0/1/0/0.2 l2transport
  encapsulation dot1q 10-100

interface g 0/2/0/0.2 l2transport
  encapsulation dot1q 10-100
```

Configuring Interconnection Node: Example

This example shows you how to configure an interconnection node. The following figure illustrates an open ring scenario.

Figure 24: Open Ring Scenario - interconnection node



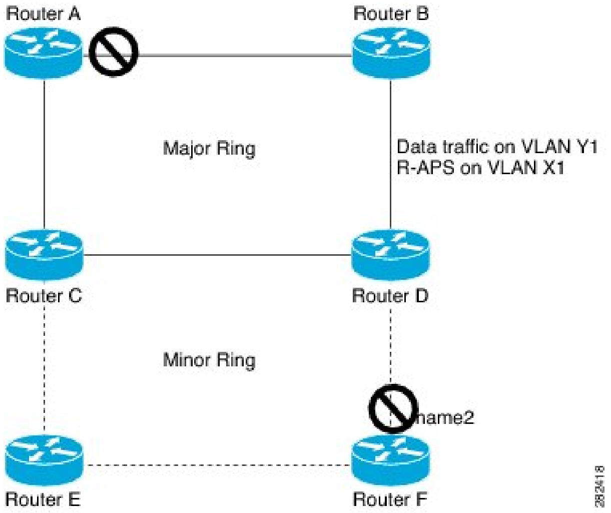
The minimum configuration required for configuring G.8032 at Router C (Open ring – Router C):

```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
interface <ifname3.10> l2transport
 encapsulation dot1q Y1
l2vpn
ethernet ring g8032 <ring-name>
      port0 interface <main port ifname1>
      port1 interface none #? This router is connected to an interconnection node
      open-ring #? Mandatory when a router is part of an open-ring
      instance <1-2>
         inclusion-list vlan-ids  X1-Y1
         aps-channel
           Port0 interface <ifname1.1>
           Port1 none #? This router is connected to an interconnection node
 bridge group bg1
    bridge-domain bd-aps#? APS-channel has its own bridge domain
       <ifname1.1> #? There is only one APS-channel at the interconnection node
    bridge-domain bd-traffic #? Data traffic has its own bridge domain
       <ifname1.10>
       <ifname2.10>
       <ifname3.10>
```

Configuring the Node of an Open Ring: Example

This example shows you how to configure the node part of an open ring. The following figure illustrates an open ring scenario.

Figure 25: Open Ring Scenario



The minimum configuration required for configuring G.8032 at the node of the open ring (node part of the open ring at router F):

```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname2.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
l2vpn
   ethernet ring g8032 <ring-name>
      port0 interface <main port ifname1>
      port1 interface <main port ifname2>
      open-ring #? Mandatory when a router is part of an open-ring
      instance <1-2>
         inclusion-list vlan-ids  X1-Y1
      rpl port1 owner #? This node is RPL owner and <main port ifname2> is blocked
         aps-channel
           port0 interface <ifname1.1>
           port1 interface <ifname2.1>
 bridge group bg1
    bridge-domain bd-aps#? APS-channel has its own bridge domain
       <ifname1.1>
       <ifname2.1>
    bridge-domain bd-traffic #? Data traffic has its own bridge domain
```

```
<ifname1.10>
<ifname2.10>
```

CHAPTER 9

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.

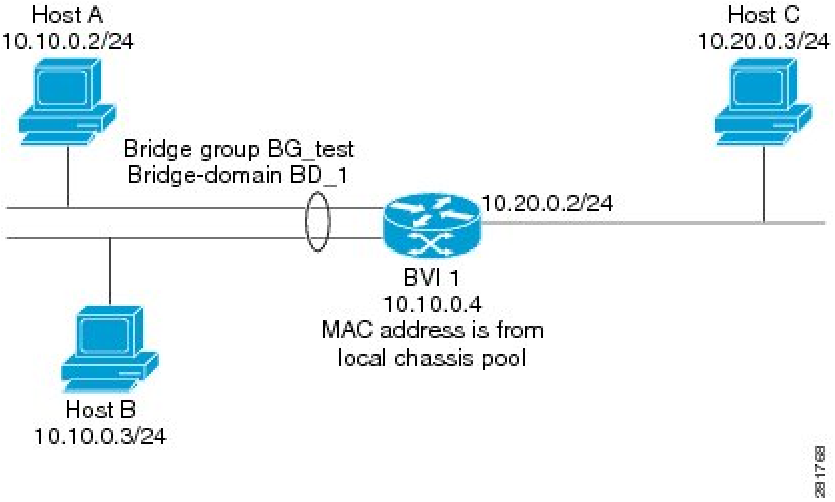
- Be sure that the BVI network address is being advertised by running static or dynamic routing on the BVI interface.

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 26:

Figure 27: 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
```

CHAPTER 10

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-legacy-bpdu** command on MST default instance, and if one of the bridge ports receives legacy BPDU, the port enters **error-disable** state.

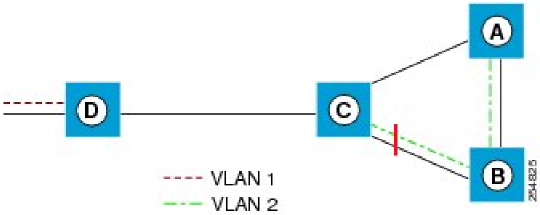
BPDU Guard

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.

Flush Containment

Flush containment is a Cisco feature that helps prevent unnecessary MAC flushes due to unrelated topology changes in other areas of a network. This is best illustrated by example. The following figure shows a network containing four devices. Two VLANs are in use: VLAN 1 is only used on device D, while VLAN 2 spans devices A, B and C. The two VLANs are in the same spanning tree instance, but do not share any links.

Figure 28: Flush Containment



If the link AB goes down, then in normal operation, as C brings up its blocked port, it sends out a topology change notification on all other interfaces, including towards D. This causes a MAC flush to occur for VLAN 1, even though the topology change which has taken place only affects VLAN 2.

Flush containment helps deal with this problem by preventing topology change notifications from being sent on interfaces on which no VLANs are configured for the MSTI in question. In the example network this would mean no topology change notifications would be sent from C to D, and the MAC flushes which take place would be confined to the right hand side of the network.

Note Flush containment is enabled by default, but can be disabled by configuration, thus restoring the behavior described in the IEEE 802.1Q standard.

Bringup Delay

Bringup delay is a Cisco feature that stops MSTP from considering an interface when calculating the spanning tree, if the interface is not yet ready to forward traffic. This is useful when a line card first boots up, as the system may declare that the interfaces on that card are *Up* before the dataplane is fully ready to forward traffic. According to the standard, MSTP considers the interfaces as soon as they are declared *Up*, and this may cause it to move other interfaces into the blocking state if the new interfaces are selected instead.

Bringup delay solves this problem by adding a configurable delay period which occurs as interfaces that are configured with MSTP first come into existence. Until this delay period ends, the interfaces remain in blocking state, and are not considered when calculating the spanning tree.

Bringup delay only takes place when interfaces which are already configured with MSTP are created, for example, on a card reload. No delay takes place if an interface which already exists is later configured with MSTP.

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
!
!
```

Configuring MSTP

This section describes the procedure for configuring MSTP:

Note This section does not describe how to configure data switching. Refer to the Implementing Multipoint Layer 2 Services module for more information.

Enabling MSTP

By default, STP is disabled on all interfaces. MSTP should be explicitly enabled by configuration on each physical or Ethernet Bundle interface. When MSTP is configured on an interface, all the subinterfaces of that interface are automatically MSTP-enabled.

Configuring MSTP parameters

The MSTP Standard defines a number of configurable parameters. The global parameters are:

- Region Name and Revision
- Bringup Delay
- Forward Delay
- Max Age or Hops
- Transmit Hold Count
- Provider Bridge mode
- Flush Containment
- VLAN IDs (per spanning-tree instance)
- Bridge Priority (per spanning-tree instance)

The per-interface parameters are:

- External port path cost

- Hello Time
- Link Type
- Port Fast and BPDU Guard
- Root Guard and Topology Change Guard
- Port priority (per spanning-tree instance)
- Internal port path cost (per spanning-tree instance)

Per-interface configuration takes place in an interface submode within the MST configuration submode.

Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default value.

SUMMARY STEPS

1. **configure**
2. **spanning-tree mst** *protocol instance identifier*
3. **bringup delay for** *interval* { **minutes** | **seconds** }
4. **flush containment disable**
5. **name** *name*
6. **revision** *revision* -number
7. **forward-delay** *seconds*
8. **maximum** { **age** *seconds* | **hops** *hops* }
9. **transmit hold-count** *count*
10. **provider-bridge**
11. **instance** *id*
12. **priority** *priority*
13. **vlan-id** *vlan-range* [,*vlan-range*][,*vlan-range*] [,*vlan-range*]
14. **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance*
15. **instance** *id* **port-priority** *priority*
16. **instance** *id* **cost** *cost*
17. **external-cost** *cost*
18. **link-type** { **point-to-point** | **multipoint** }
19. **hello-time** *seconds*
20. **portfast** [**bpdu-guard**]
21. **guard root**
22. **guard topology-change**
23. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters the XR Config mode.

Step 2 **spanning-tree mst** *protocol instance identifier*

Example:

```
RP/0/RP0/CPU0:router(config)# spanning-tree mst a
RP/0/RP0/CPU0:router(config-mstp)#
```

Enters the MSTP configuration submode.

Step 3 **bringup delay for** *interval* { **minutes** | **seconds** }

Example:

```
RP/0/RP0/CPU0:router(config-mstp)#bringup delay for 10 minutes
```

Configures the time interval to delay bringup for.

Step 4 **flush containment disable**

Example:

```
RP/0/RP0/CPU0:router(config-mstp)#flush containment disable
```

Disable flush containment.

This command performs MAC flush on all instances regardless of the their state.

Step 5 **name** *name*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# name m1
```

Sets the name of the MSTP region.

The default value is the MAC address of the switch, formatted as a text string by means of the hexadecimal representation specified in IEEE Std 802.

Step 6 **revision** *revision* -number

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# revision 10
```

Sets the revision level of the MSTP region.

Allowed values are from 0 through 65535.

Step 7 **forward-delay** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# forward-delay 20
```

Sets the forward-delay parameter for the bridge.

Allowed values for bridge forward-delay time in seconds are from 4 through 30.

Step 8 **maximum** { **age** *seconds* | **hops** *hops* }

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# max age 40
RP/0/RP0/CPU0:router(config-mstp)# max hops 30
```

Sets the maximum age and maximum hops performance parameters for the bridge.

Allowed values for maximum age time for the bridge in seconds are from 6 through 40.

Allowed values for maximum number of hops for the bridge in seconds are from 6 through 40.

Step 9 **transmit hold-count** *count*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# transmit hold-count 8
```

Sets the transmit hold count performance parameter.

Allowed values are from 1 through 10.

Step 10 **provider-bridge**

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# provider-bridge
```

Places the current instance of the protocol in 802.1ad mode.

Step 11 **instance** *id*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# instance 101
RP/0/RP0/CPU0:router(config-mstp-inst)#
```

Enters the MSTI configuration submode.

Allowed values for the MSTI ID are from 0 through 4094.

Step 12 **priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-inst)# priority 8192
```

Sets the bridge priority for the current MSTI.

Allowed values are from 0 through 61440 in multiples of 4096.

Step 13 **vlan-id** *vlan-range* [,*vlan-range*][,*vlan-range*] [,*vlan-range*]

Example:

```
RP/0/RP0/CPU0:router(config-mstp-inst)# vlan-id 2-1005
```

Associates a set of VLAN IDs with the current MSTI.

List of VLAN ranges in the form a-b, c, d, e-f, g, and so on.

Note Repeat steps 11 to 13 for each MSTI.

Step 14 **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance*

Example:

```
RP/0/RP0/CPU0:router(config-mstp)# interface FastEthernet 0/0/0/1
RP/0/RP0/CPU0:router(config-mstp-if)#
```

Enters the MSTP interface configuration submode, and enables STP for the specified port.

Forward interface in Rack/Slot/Instance/Port format.

Step 15 **instance** *id* **port-priority** *priority*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# instance 101 port-priority 160
```

Sets the port priority performance parameter for the MSTI.

Allowed values for the MSTI ID are from 0 through 4094.

Allowed values for port priority are from 0 through 240 in multiples of 16.

Step 16 **instance** *id* **cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# instance 101 cost 10000
```

Sets the internal path cost for a given instance on the current port.

Allowed values for the MSTI ID are from 0 through 4094.

Allowed values for port cost are from 1 through 200000000.

Repeat steps 15 and 16 for each MSTI for each interface.

Step 17 **external-cost** *cost*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# external-cost 10000
```

Sets the external path cost on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 18 **link-type** { **point-to-point** | **multipoint** }

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# link-type point-to-point
```

Sets the link type of the port to point-to-point or multipoint.

Step 19 **hello-time** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are 1 and 2.

Step 20 **portfast** [**bpdu-guard**]

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# portfast
RP/0/RP0/CPU0:router(config-mstp-if)# portfast bpduguard
```

Enables PortFast on the port, and optionally enables BPDU guard.

Step 21 **guard root**

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# guard root
```

Enables RootGuard on the port.

Step 22 **guard topology-change**

Example:

```
RP/0/RP0/CPU0:router(config-mstp-if)# guard topology-change
```

Enables TopologyChangeGuard on the port.

Note Repeat steps 14 to 22 for each interface.

Step 23 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Verifying MSTP

These show commands allow you to verify the operation of MSTP:

- **show spanning-tree mst** *mst-name*
- **show spanning-tree mst** *mst-name* **interface** *interface-name*
- **show spanning-tree mst** *mst-name* **errors**
- **show spanning-tree mst** *mst-name* **configuration**
- **show spanning-tree mst** *mst-name* **bpdu interface** *interface-name*
- **show spanning-tree mst** *mst-name* **topology-change flushes**

Configuring MSTP: Examples

This example shows MSTP configuration for a single spanning-tree instance with MSTP enabled on a single interface:

```
config
spanning-tree mst customer1
name customer1
revision 1
instance 0
  priority 28672
!
instance 1
  vlan-ids 1001
  priority 28672
!
interface bundle-ether8171
!
interface bundle-ether861

interface Bundle-Ether861.10000 l2transport
encapsulation dot1q 1001
!
interface Bundle-Ether861.10001 l2transport
encapsulation dot1q 1002

interface Bundle-Ether8171.10000 l2transport
encapsulation dot1q 1001
!
interface Bundle-Ether8171.10001 l2transport
encapsulation dot1q 1002
```

This example shows the output from the **show spanning-tree mst** command, which produces an overview of the spanning tree protocol state:

```
show spanning-tree mst customer1
Role:  ROOT=Root, DSGN=Designated, ALT=Alternate, BKP=Backup, MSTR=Master
State: FWD=Forwarding, LRN=Learning, BLK=Blocked, DLY=Bringup Delayed
```

```
Operating in dot1q mode


MSTI 0 (CIST):

  VLANS Mapped: 1-1000,1006-4094

  CIST Root  Priority    24576
             Address     b026.80da.e800
             Ext Cost    0

  Root ID    Priority    24576
             Address     b026.80da.e800
             Int Cost    10000
             Max Age 20 sec, Forward Delay 15 sec


  Bridge ID  Priority    28672 (priority 28672 sys-id-ext 0)
             Address     00bc.6025.64d8
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6


Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
BE8171       128.2   10000     ALT  BLK   28672 14a2.a05c.6600 128.1
BE861        128.1   10000     ROOT FWD   24576 b026.80da.e800 128.1


MSTI 1:

  VLANS Mapped: 1001-1005

  Root ID    Priority    24576
             Address     14a2.a05c.6600
             Int Cost    10000
             Max Age 20 sec, Forward Delay 15 sec


  Bridge ID  Priority    28672 (priority 28672 sys-id-ext 0)
             Address     00bc.6025.64d8
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6


Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
BE8171       128.2   10000     ROOT FWD   24576 14a2.a05c.6600 128.1
BE861        128.1   10000     ALT  BLK   24576 b026.80da.e800 128.1
```

In the **show spanning-tree mst** example output, the first line indicates whether MSTP is operating in dot1q or the Provider Bridge mode, and this information is followed by details for each MSTI.

For each MSTI, the following information is displayed:

- The list of VLANs for the MSTI.
- For the CIST, the priority and bridge ID of the CIST root, and the external path cost to reach the CIST root. The output also indicates if this bridge is the CIST root.
- The priority and bridge ID of the root bridge for this MSTI, and the internal path cost to reach the root. The output also indicates if this bridge is the root for the MSTI.

- The max age and forward delay times received from the root bridge for the MSTI.
- The priority and bridge ID of this bridge, for this MSTI.
- The maximum age, forward delay, max hops and transmit hold-count for this bridge (which is the same for every MSTI).
- A list of MSTP-enabled interfaces. For each interface, the following information is displayed:
  - The interface name
  - The port priority and port ID for this interface for this MSTI.
  - The port cost for this interface for this MSTI.
  - The current port role:
    - DSGN—Designated: This is the designated port on this LAN, for this MSTI
    - ROOT—Root: This is the root port for the bridge for this MSTI.
    - ALT—Alternate: This is an alternate port for this MSTI.
    - BKP—Backup: This is a backup port for this MSTI
    - MSTR—Master: This is a boundary port that is a root or alternate port for the CIST.

    The interface is down, or the bringup delay timer is running and no role has been assigned yet.
  - The current port state:
    - BLK—The port is blocked.
    - LRN—The port is learning.
    - FWD—The port is forwarding.
    - DLY—The bringup-delay timer is running.
  - If the port is a boundary port, and not CIST and the port is not designated, then only the BOUNDARY PORT is displayed and the remaining information is not displayed.
  - If the port is not up, or the bringup delay timer is running, no information is displayed for the remaining fields. Otherwise, the bridge priority and bridge ID of the designated bridge on the LAN that the interface connects to is displayed, followed by the port priority and port ID of the designated port on the LAN. If the port role is Designated, then the information for this bridge or port is displayed.

This example shows the output of show spanning-tree mst , which displays details about the topology changes that have occurred for each MSTI on each interface:

```
spanning-tree mst customer1 topology-change flushes
STI 0 (CIST):
Interface     Last TC              Reason                           Count
------------  -------------------- -------------------------------- -----
BE8171        23:05:36 Jun  6 2023  Role change: ROOT to ALT            1
BE861         --------------------  No flushes                          0


MSTI 1:
```

```
Interface     Last TC              Reason                           Count
------------  -------------------- -------------------------------- -----
BE8171        -------------------- No flushes                           0
BE861         -------------------- Flush Containment active         -----
```

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:

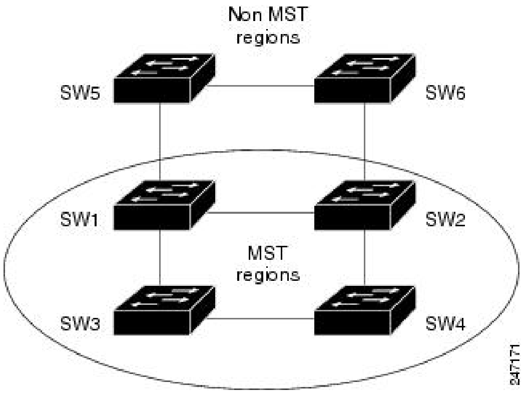
- Legacy STP (STP)—The original STP protocol was defined in IEEE 802.1D-1998. This creates a single spanning tree which is used for all VLANs and most of the convergence is timer-based.
- Multiple STP (MSTP)—A further enhancement was defined in IEEE 802.1Q-2005. This allows multiple spanning tree instances 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.

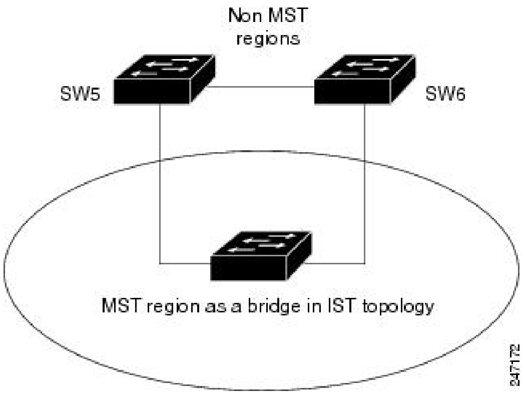
The following figure shows the operation of MST regions when bridges running MSTP are connected to bridges running legacy STP or RSTP. In this example, switches SW1, SW2, SW3, SW4 support MSTP, while switches SW5 and SW6 do not.

Figure 29: 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 30: 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.