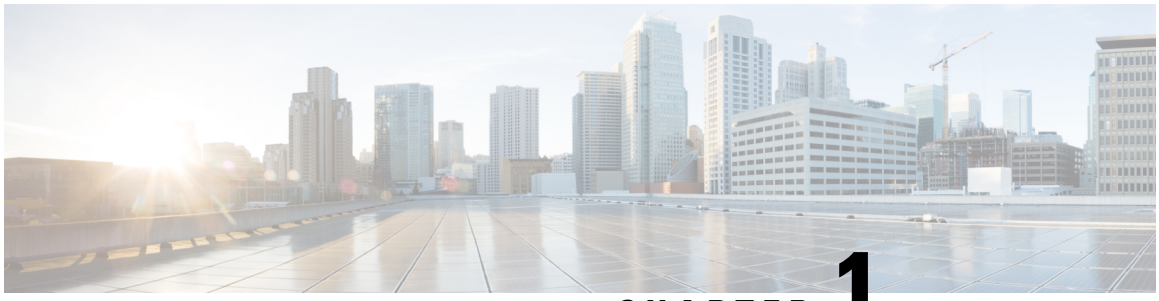
MPLS Layer 2 VPNs Configuration Guide, Cisco IOS XE 17 (NCS 4200 Series)

First Published: 2019-12-21

Last Modified: 2021-04-07

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

CONTENTS



CHAPTER 1

Feature History

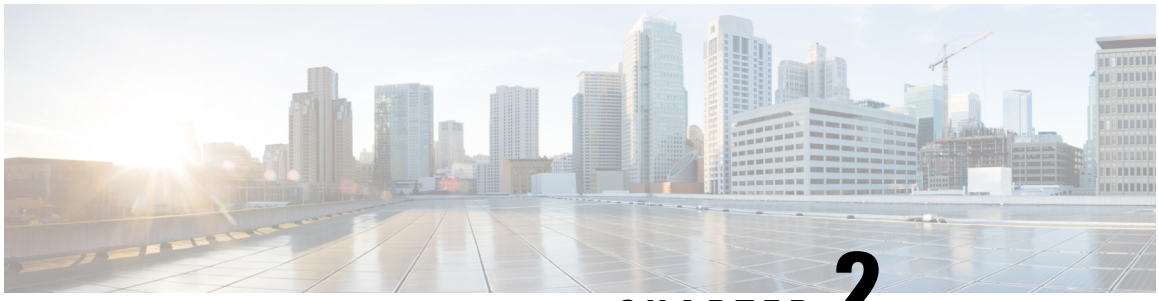
The following table lists the new and modified features supported in the MPLS Layer 2 VPNs Configuration Guide in Cisco IOS XE 17 releases, on Cisco NCS 4201 and Cisco NCS 4202 routers.

Feature	Description
Cisco IOS XE Bengaluru 17.6.1	
Remote LFA for MLDP	Remote Loop-Free Alternate (RLFA) based Fast Reroute (FRR) improves LFA coverage. When used with Multicast Label Distribution Protocol (MLDP) for IPv4, there is no need for an extra protocol in the control plane.
Cisco IOS XE Bengaluru 17.5.1	
On-Change Notifications for L2VPN Pseudowire	This feature allows you to subscribe on-change Network Configuration Protocol (NETCONF) notifications for L2VPN pseudowire. You can generate an alert from a device when the pseudowire status changes.
Cisco IOS XE Amsterdam 17.3.1	
EVPN Single-Homing Over MPLS for NCS 4201 and NCS 4202	The EVPN Single-Homing feature utilizes the BGP MPLS-based Ethernet VPN functionality as defined in RFC 7432. That is, to achieve single-homing between a Provider Edge (PE) and a Customer Edge (CE) device. There are three fundamental building blocks for EVPN technology, EVPN Instance (EVI), Ethernet Segment (ES), EVPN BGP routes and extended communities. For EVPN Single-Homing feature, a CE device is attached to a single PE device and has an Ethernet Segment.

The following table lists the new and modified features supported in the MPLS Layer 2 VPNs Configuration Guide in Cisco IOS XE 17 releases, on Cisco NCS 4206 and Cisco NCS 4216 routers.

Feature	Description
Cisco IOS XE Bengaluru 17.6.1	
Remote LFA for MLDP	Remote Loop-Free Alternate (RLFA) based Fast Reroute (FRR) improves LFA coverage. When used with Multicast Label Distribution Protocol (MLDP) for IPv4, there is no need for an extra protocol in the control plane.
Cisco IOS XE Bengaluru 17.5.1	

Feature	Description
On-Change Notifications for L2VPN Pseudowire	This feature allows you to subscribe on-change Network Configuration Protocol (NETCONF) notifications for L2VPN pseudowire. You can generate an alert from a device when the pseudowire status changes.
Cisco IOS XE Amsterdam 17.1.1	
EVPN Single-Homing Over MPLS for NCS 4206 and NCS 4216	The EVPN Single-Homing feature utilizes the BGP MPLS-based Ethernet VPN functionality as defined in RFC 7432. That is, to achieve single-homing between a Provider Edge (PE) and a Customer Edge (CE) device. There are three fundamental building blocks for EVPN technology, EVPN Instance (EVI), Ethernet Segment (ES), EVPN BGP routes and extended communities. For EVPN Single-Homing feature, a CE device is attached to a single PE device and has an Ethernet Segment.



CHAPTER 2

L2VPN Protocol-Based CLIs

The L2VPN Protocol-Based CLIs feature provides a set of processes and an improved infrastructure for developing and delivering Cisco IOS software on various Cisco platforms. This feature introduces new commands and modifies or replaces existing commands to achieve a consistent functionality across Cisco platforms and provide cross-Operating System (OS) support.

Information About L2VPN Protocol-Based CLIs

Overview of L2VPN Protocol-Based CLIs

The L2VPN Protocol-Based CLIs feature introduces new commands and modifies or replaces existing commands to achieve a consistent functionality across Cisco platforms and provide cross-Operating System (OS) support.

Note The new, updated, and replacement commands are available in Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S. However, the legacy commands that are being replaced will be deprecated in later releases.

Benefits of L2VPN Protocol-Based CLIs

The L2VPN Protocol-Based CLIs feature provides the following benefits:

- Consistent user experience across different operating systems.
- Consistent configuration for all Layer 2 VPN (L2VPN) scenarios.
- Enhanced functionality that is achieved by configuring pseudowires as virtual interfaces and monitoring the pseudowires as physical ports.
- Feature configuration such as quality of service (QoS) service policies on individual pseudowires .
- Redundant pseudowire configuration that is independent of the primary pseudowire to provide enhanced high availability.

These benefits are achieved through the following enhancements:

- New service contexts can be created for point-to-point and multipoint Layer 2 services by using the new L2VPN cross connect and L2VPN virtual forwarding interface (VFI) contexts.
  - The L2VPN cross connect context is used for configuring point-to-point pseudowires, pseudowire stitching, and local switching (hair pinning). Ethernet interfaces and subinterfaces, Ethernet Flow Points (EFP), ATM interfaces and WAN interfaces (PPP,HDLC,Serial), and pseudowire interfaces can be defined as members of an L2VPN cross connect context.
  - The L2VPN VFI context instantiates Virtual Private LAN Services (VPLS) VFI for multipoint scenarios. Pseudowires can be defined as members of an L2VPN VFI context.
  - Bridge domains or VLANs are used for multipoint scenarios. EFPs, pseudowires, or VFIs can be configured as members of a bridge domain. Pseudowires can be configured as member of a VFI. The VFI can be configured as a member of a VLANbridge domains.
- New port contexts can be created (dynamically or manually) for pseudowires by using the pseudowire interface.
- Pseudowire customization can be achieved using interface templates and pseudowire interfaces that are applied to L2VPN context members. Pseudowire customizations include following features:
  - Encapsulation type
  - Control word
  - Maximum Transmission Unit (MTU)
  - Pseudowire signaling type
  - Tunnel selection
- Interworking and redundancy group service attributes can be configured under the L2VPN service context. The redundancy groups are configured independently from the primary pseudowire, which helps achieve zero traffic interruptions while adding, modifying, or deleting backup pseudowires.

L2VPN Protocol-Based CLI Changes

The following commands are introduced in Cisco IOS XE Release 3.7S, Cisco IOS Release 15.3(1)S, and Cisco IOS Release 15.4(1)S:

- **debug l2vpn pseudowire**
- **l2vpn**
- **l2vpn pseudowire static-oam class**
- **monitor event-trace l2vpn**
- **show interface pseudowire**
- **show l2vpn service**
- **shutdown (MPLS)**
- **vc**

The following commands are modified in Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S:

- **auto-route-target**
- **bridge-domain parameterized vlan**
- **debug condition xconnect fib**
- **debug condition xconnect interface**
- **debug condition xconnect peer**
- **debug condition xconnect segment**
- **description**
- **encapsulation (MPLS)**
- **forward permit l2protocol all**
- **interworking**
- **l2vpn subscriber authorization group**
- **l2vpn xconnect context**
- **load-balance flow**
- **monitor event-trace ac**
- **monitor event-trace atom**
- **monitor event-trace l2tp**
- **monitor peer bfd**
- **mtu**
- **preferred-path**
- **remote circuit id**
- **rd (VPLS)**
- **route-target (VPLS)**
- **sequencing**
- **status**
- **status admin-down disconnect**
- **status control-plane route-watch**
- **status decoupled**
- **status peer topology dual-homed**
- **status protocol notification static**
- **status redundancy**
- **switching tlv**

- **tlv**
- **tlv template**
- **vccv**
- **vccv bfd status signaling**
- **vccv bfd template**
- **vpls-id**
- **vpn id (MPLS)**

The table below lists the legacy commands that will be replaced in future releases. From Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S both new and legacy commands will coexist until the legacy commands are deprecated in future releases.

Table 1: Replacement Commands Introduced in Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S

Legacy Command	Replacement Command Introduced in Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S
backup delay	**redundancy delay (under l2vpn xconnect context)**
bridge-domain (service instance)	**member (bridge-domain)**
clear mpls l2transport fsm state transition	**clear l2vpn atom fsm state transition**
clear mpls l2transport fsm event	**clear l2vpn atom fsm event**
clear xconnect	**clear l2vpn service**
connect (L2VPN local switching)	**l2vpn xconnect context**
debug acircuit	**debug l2vpn acircuit**
debug mpls l2transport checkpoint	**debug l2vpn atom checkpoint**
debug mpls l2transport event-trace	**debug l2vpn atom event-trace**
debug mpls l2transport fast-failure-detect	**debug l2vpn atom fast-failure-detect**
debug mpls l2transport signaling	**debug l2vpn atom signaling**
debug mpls l2transport static-oam	**debug l2vpn atom static-oam**
debug mpls l2transport vc subscriber	**debug l2vpn atom vc**
debug mpls l2transport vc	**debug l2vpn atom vc**
debug mpls l2transport vc vccv bfd event	**debug l2vpn atom vc vccv**
debug vfi	**debug l2vpn vfi**
debug vfi checkpoint	**debug l2vpn vfi checkpoint**

Legacy Command	Replacement Command Introduced in Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S
debug xconnect	**debug l2vpn xconnect**
debug xconnect rib	**debug l2vpn xconnect rib**
description (L2VFI)	**description (L2VPN)**
l2 pseudowire routing	**pseudowire routing**
l2 router-id	**router-id**
l2 vfi	**l2vpn vfi context**
l2 subscriber	**l2vpn subscriber**
l2 vfi autodiscovery	**autodiscovery**
l2 vfi point-to-point	**l2vpn xconnect context**
local interface	**pseudowire type**
monitor event-trace st-pw-oam	**monitor event-trace pwoam**
mpls label	**label (pseudowire)**
mpls control-word	**control-word (encapsulation mpls under l2vpn connect context)**
neighbor (l2 vfi)	**member (l2vpn vfi)**
protocol	**signaling protocol**
pseudowire-static-oam class	**l2vpn pseudowire static-oam class**
pseudowire tlv template	**l2vpn pseudowire tlv template**
pw-class keyword in the **xconnect** command	**source template type pseudowire**
remote link failure notification	**l2vpn remote link failure notification**
show mpls l2transport binding	**show l2vpn atom binding**
show mpls l2transport checkpoint	**show l2vpn atom checkpoint**
show mpls l2transport hw-capability	**show l2vpn atom hw-capability**
show mpls l2transport static-oam	**show l2vpn atom static-oam**
show mpls l2transport summary	**show l2vpn atom summary**
show mpls l2transport pwid	**show l2vpn atom pwid**
show mpls l2transport vc	**show l2vpn atom vc**

Legacy Command	Replacement Command Introduced in Cisco IOS XE Release 3.7S and Cisco IOS Release 15.3(1)S
show xconnect pwmib	**show l2vpn pwmib**
show xconnect rib	**show l2vpn rib**
show xconnect	**show l2vpn service**
show vfi	**show l2vpn vfi**
xconnect	**l2vpn xconnect context** and **member**
xconnect logging pseudowire status global	**logging pseudowire status**
xconnect logging redundancy global	**logging redundancy**
xconnect *peer-ip vc-id*	**neighbor peer-ip vc-id (xconnect context)**

MPLS L2VPN Protocol-Based CLI: Examples

The examples in this section provide the new configurations that are introduced by the MPLS L2VPN Protocol-Based CLIs feature that replace the existing (legacy) MPLS L2VPN CLIs.

MPLS L2VPN VPWS Configuration Using Replacement (or New) Commands

The following example shows the configuration for Virtual Private Wired Service (VPWS)—Ethernet over Multiprotocol Label Switching (EoMPLS). In this example, L2VPN members point to peer ID or virtual circuit (VC) ID. This configuration is used in most cases except when features like quality of service (QoS), need to be applied at the pseudowire level.

```
l2vpn xconnect context foo
  member GigabitEthernet2/1/1 service-instance 300
  member 10.0.0.1 888 encapsulation mpls
!
interface GigabitEthernet2/1/1
  service instance 300 GigabitEthernetEthernet
   encapsulation dot1q 30
   rewrite ingress tag pop 1 symmetric
  !
  service instance 400 GigabitEthernetEthernet
   encapsulation dot1q 40
   rewrite ingress tag pop 1 symmetric

l2vpn xconnect context faa
  member GigabitEthernet2/1/1 service-instance 400
  member 10.0.0.1 999 encapsulation mpls
!
```

MPLS L2VPN Pseudowire Configuration Using Replacement (or New) Commands

In the following example, L2VPN members point to a pseudowire interface. The pseudowire interface is manually configured and includes peer ID and VC ID. This configuration is used in most cases except when features like quality of service (QoS), need to be applied at the pseudowire level.

```
l2vpn xconnect context foo
  member GigabitEthernet2/1/1 service-instance 300
  member Pseudowire888
!
```

```
interface Pseudowire 888
 encapsulation mpls
 neighbor 10.0.0.1 888
!
interface Pseudowire 999
 encapsulation mpls
 neighbor 10.0.0.1 999
!
interface GigabitEthernet2/1/1
  service instance 300 GigabitEthernetEthernet
   encapsulation dot1q 30
   rewrite ingress tag pop 1 symmetric
  !
  service instance 400 GigabitEthernetEthernet
   encapsulation dot1q 40
   rewrite ingress tag pop 1 symmetric

l2vpn xconnect context faa
  member GigabitEthernet2/1/1 service-instance 400
  member Pseudowire 999
!
```

MPLS L2VPN Pseudowire Redundancy Configuration Using Replacement (or New) Commands

The following example shows the configuration for pseudowire redundancy. The new configuration shows concise pseudowire redundancy with no submodes or separate groups. This configuration allows the addition of redundant members to a service without service disruption. This configuration also allows modifying or deleting redundant service configurations without service disruption.

```
l2vpn xconnect context sample-pw-redundancy
  member Ethernet2/1GigabitEthernet2/1/1 service-instance 200
  member 10.1.1.1 180 encap mpls group Denver
  member 2.2.2.2 180180 encap mpls group Denver priority 1
  member 3.3.3.3 180181 encap mpls group Denver priority 2
  redundancy delay 1 20 group Denver
!
interface GigabitEthernet2/1/1
  service instance 200 GigabitEthernetEthernet
   encapsulation dot1q 100
   rewrite ingress tag pop 1 symmetric
```

MPLS L2VPN Static Pseudowire Configuration Using Replacement (or New) Commands

Note The following configuration is shown for the Provider Edge (PE) 1 router in a network scheme where Customer Edge (CE) 1 and PE 1 and PE 2 and CE 2 traverse through a Provider core (P) router (CE 1—PE 1—P—PE 2—CE 2).

```
interface g2/1/1
service instance 300 ethernet
encapsulation dot1q 300
no shutdown
!
interface pseudowire 100
neighbor 10.4.4.4 121
encapsulation mpls
label 200 300
signaling protocol none
no shutdown
!
l2vpn xconnect context foo
```

```
  member GigabitEthernet2/1/1 service-instance 300
  member pseudowire 100
```

MPLS L2VPN Static Pseudowire Template Configuration Using Replacement (or New) Commands

Note The following configuration is shown for the Provider Edge (PE) 1 router in a network scheme where Customer Edge (CE) 1 and PE 1 and PE 2 and CE 2 traverse through a Provider core (P) router (CE 1—PE 1—P—PE 2—CE 2).

```
template type pseudowire test
encapsulation mpls
signaling protocol none
!
interface g2/1/1
service instance 300 ethernet
encapsulation dot1q 300
no shutdown
!
interface pseudowire 100
neighbor 10.4.4.4 121
source template type pseudowire test
label 200 300
no shutdown
!
l2vpn xconnect context foo
  member GigabitEthernet2/1/1 service-instance 300
  member pseudowire 100
```

MPLS L2VPN Dynamic Pseudowire Template Configuration Using Replacement (or New) Commands

Note The following configuration is shown for the Provider Edge (PE) 1 router in a network scheme where Customer Edge (CE) 1 and PE 1 and PE 2 and CE 2 traverse through a Provider core (P) router (CE 1—PE 1—P—PE 2—CE 2).

```
template type pseudowire test
encapsulation mpls
signaling protocol ldp
!
!
interface g2/1/1
service instance 300 ethernet
encapsulation dot1q 300
no shutdown
!
interface pseudowire 100
neighbor 10.4.4.4 121
source template type pseudowire test
no shutdown
!
l2vpn xconnect context foo
  member GigabitEthernet2/1/1 service-instance 300
  member pseudowire 100
```

MPLS L2VPN Multi-segment Static-Dynamic Pseudowire Template Configuration Using Replacement (or New) Commands

The following PE router configuration is for a multi-segment static-dynamic pseudowire:

```
l2vpn pseudowire tlv template TLV
 tlv mtu 1 4 dec 1500
!
interface pseudowire401
  source template type pseudowire staticTempl
encapsulation mpls
neighbor 10.4.4.4 101
signaling protocol none
label 4401 4301
pseudowire type 4
tlv template TLV
tlv 1 4 dec 1500
tlv vccv-flags C 4 hexstr 0110
!
interface pseudowire501
  source template type pseudowire dynTempl
encapsulation mpls
neighbor 10.2.2.2 101
signaling protocol ldp
```

Displaying MPLS L2VPN Pseudowire Template Configuration Using Replacement (or New) Commands

The following example displays output from the **show interface pseudowire** command:

```
PE1#show interface pseudowire 100
pseudowire100 is up
    Description: Pseudowire Interface
    MTU 1500 bytes, BW 10000000 Kbit
    Encapsulation mpls
    Peer IP 10.4.4.4, VC ID 121
    RX
      21 packets 2623 bytes 0 drops
    TX
      20 packets 2746 bytes 0 drops
```

The following example displays output from the **show template** command:

```
PE1#show template

Template        class/type    Component(s)
ABC             owner        interface pseudowire
 BOUND: pw1
```

Sourcing a Template Under an Interface Pseudowire Using Replacement (or New) Commands

The following example configures the interface pseudowire to inherit all attributes defined from a template on the PE 2 router.

```
PE2(config-subif)#interface pseudowire 100
PE2(config-if)#source template type pseudowire test
PE2(config-if)#neighbor 10.4.4.4 121
PE2(config-if)#no shutdown
```

Additional References

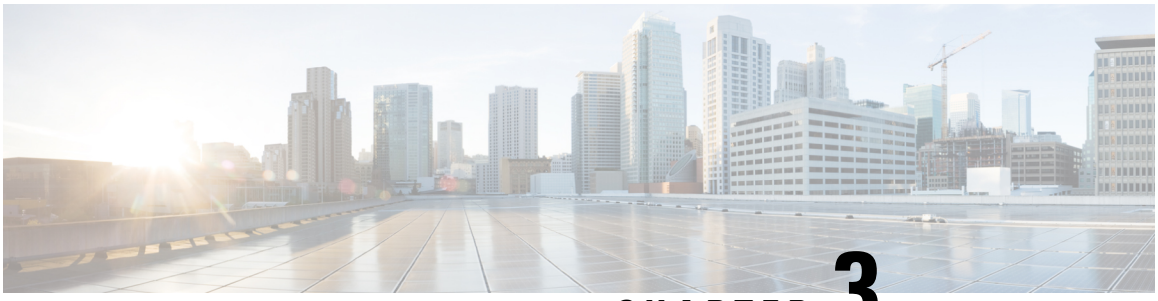
Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases

Related Topic	Document Title
MPLS commands	Multiprotocol Label Switching Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 3

Any Transport over MPLS

This module describes how to configure Any Transport over MPLS (AToM) transports data link layer (Layer 2) packets over a Multiprotocol Label Switching (MPLS) backbone. AToM enables service providers to connect customer sites with existing Layer 2 networks by using a single, integrated, packet-based network infrastructure--a Cisco MPLS network. Instead of using separate networks with network management environments, service providers can deliver Layer 2 connections over an MPLS backbone. AToM provides a common framework to encapsulate and transport supported Layer 2 traffic types over an MPLS network core.

AToM supports the following like-to-like transport types:

- ATM Adaptation Layer Type-5 (AAL5) over MPLS
- ATM Cell Relay over MPLS
- Ethernet over MPLS (VLAN and port modes)
- Circuit Emulation (CEM)
- Frame Relay over MPLS
- PPP over MPLS
- High-Level Data Link Control (HDLC) over MPLS

Note For information on ATM Cell relay and Circuit Emulation(CEM), see Configuring Pseudowire.

Prerequisites for Any Transport over MPLS

- IP routing must be configured in the core so that the provider edge (PE) routers can reach each other via IP.
- MPLS must be configured in the core so that a label-switched path (LSP) exists between the PE routers.
- Cisco Express Forwarding must be enabled before you configure any Layer 2 circuits.
- A loopback interface must be configured for originating and terminating Layer 2 traffic. Ensure that the PE routers can access the other router's loopback interface. Note that the loopback interface is not needed in all cases. For example, tunnel selection does not need a loopback interface when AToM is directly mapped to a traffic engineering (TE) tunnel.
- Before converting an interface with L2TPv3 xconnect to AToM xconnect, remove the L2TPv3 configuration from the interface and then configure AToM.
- Before configuring Ethernet over MLS in VLAN mode, you must configure Ethernet over MPLS on the subinterfaces.

General Restrictions

- In a member configuration, the **l2vpn xconnect context** command does not prompt any error or warning, if you specify without a service instance.
- The **show mpls l2transport vc** <*vcid*> **detail** command output displays few LDP-related information, even in case of static pseudowire.
- Address format--Configure the Label Distribution Protocol (LDP) router ID on all PE routers to be a loopback address with a /32 mask. Otherwise, some configurations might not function properly.
- For PTPoIP configuration with explicit Null MPLS encapsulation, when a Transparent Clock (TC) is placed between a PTP primary and a PTP subordinate, the TC does not update the correction field.
- Load balancing for Layer 2 VPN traffic on a Provider router is not supported on the RSP2 Module.
- Layer 2 virtual private networks (L2VPN) features (AToM and Layer 2 Tunnel Protocol Version 3 (L2TPv3) are not supported on an ATM interface.
- Some features may not work if AToM is configured and L2TPv3 configuration is not removed properly.
- Ethernet over MPLS (EoMPLS) VC statistics are not supported on the Cisco RSP3 module.
- Virtual Circuit (VC) counters are not supported on the Cisco RSP3 module.

Note VC counters are enabled by default.

- 4000 virtual circuits are supported on the Cisco RSP3 module.
- TE-FRR with BGP labels for layer 2 and layer 3 VPNs must terminate on the BGP gateway because of the four-label limitation.

- If an AToM tunnel spans different service providers that exchange MPLS labels using IPv4 Border Gateway Protocol (BGP) (RFC 3107), you add a label to the stack. The maximum MPLS label stack is five (FRR label, TE label, LDP label, VPN label, VC label)four (FRR label, TE label, LDP label, VC label).
- BGP PIC Edge with EoMPLS using BGP label Unicast (RFC 3107) requires the **bgp mpls-local-label** command to be explicitly enabled under the Router BGP process. This limitation is applicable only on the Cisco RSP3 module.
- Hot standby pseudowire (HSPW) convergence without pseudowire grouping increments linearly. For example, for a thousand virtual circuits, it requires about 54 seconds of convergence time. This is applicable only for the Cisco RSP3 Module.

  Clear interface is not the recommended way to measure the convergence numbers.
- With two ECMP paths, load sharing on L2VPN traffic occurs based on odd or even MPLS VC labels. If L2VPN circuits have either odd **or** even MPLS VC labels, load sharing is not performed. But if L2VPN circuits have a combination of both odd **and** even MPLS VC labels, then the odd MPLS VC labels circuits select one link whereas the even MPLS VC labels circuits select another link.
- Flow-Aware Transport (FAT) Load Balancing over VPLS is not supported.

ATM AAL5 over MPLS Restrictions

- AAL5 over MPLS is supported only in SDU mode.

Ethernet over MPLS (EoMPLS) Restrictions

- The subinterfaces between the CE and PE routers that are running Ethernet over MPLS must be in the same subnet.
- The subinterface on the adjoining CE router must be on the same VLAN as the PE router.
- Ethernet over MPLS supports VLAN packets that conform to the IEEE 802.1Q standard. The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames. The Inter-Switch Link (ISL) protocol is not supported between the PE and CE routers.
- The AToM control word is supported. However, if the peer PE does not support a control word, the control word is disabled.
- Ethernet packets with hardware-level cyclic redundancy check (CRC) errors, framing errors, and runt packets are discarded on input.

Tunnel Selection Restrictions

- The selected path should be an LSP destined to the peer PE router.
- The selected tunnel must be an MPLS TE tunnel.

- If you specify an IP address, that address must be the IP address of the loopback interface on the remote PE router. The address must have a /32 mask. There must be an LSP destined to that selected address. The LSP need not be a TE tunnel.

Remote Ethernet Port Shutdown Restrictions

This feature is not symmetrical if the remote PE router is running an older version image or is on another platform that does not support the EoMPLS remote Ethernet port shutdown feature and the local PE is running an image which supports this feature.

Remote Ethernet Port Shutdown is supported only on EFP with encapsulation default.

Information About Any Transport over MPLS

To configure AToM, you must understand the following concepts:

How AToM Transports Layer 2 Packets

AToM encapsulates Layer 2 frames at the ingress PE and sends them to a corresponding PE at the other end of a pseudowire, which is a connection between the two PE routers. The egress PE removes the encapsulation and sends out the Layer 2 frame.

The successful transmission of the Layer 2 frames between PE routers is due to the configuration of the PE routers. You set up the connection, called a pseudowire, between the routers. You specify the following information on each PE router:

- The type of Layer 2 data that will be transported across the pseudowire, such as Ethernet, Frame Relay, or ATM
- The IP address of the loopback interface of the peer PE router, which enables the PE routers to communicate
- A unique combination of peer PE IP address and VC ID that identifies the pseudowire

The following example shows the basic configuration steps on a PE router that enable the transport of Layer 2 packets. Each transport type has slightly different steps.

Step 1 defines the interface or subinterface on the PE router:

```
Router# interface
 interface-type interface-number
```

Step 2 configures an ethernet service instance on an interface and enters service instance configuration mode:

```
Router(config-if)#service instance number ethernet WORD
Router(config-if)# service instance 393 ethernet ethernet1
```

Step 2 3 specifies the encapsulation type for the interface, such as dot1q:

```
Router(config-if-srv)# encapsulation
encapsulation-type
```

Step 4 does the following:

- Makes a connection to the peer PE router by specifying the LDP router ID of the peer PE router.
- Specifies a 32-bit unique identifier, called the VC ID, which is shared between the two PE routers.

The combination of the peer router ID and the VC ID must be unique on the router. Two circuits cannot use the same combination of peer router ID and VC ID.

- Specifies the tunneling method used to encapsulate data in the pseudowire. AToM uses MPLS as the tunneling method.

```
Router(config-if-srv)# xconnect
peer-router-id vcid
encapsulation mpls
```

As an alternative, you can set up a pseudowire class to specify the tunneling method and other characteristics. For more information, see the Configuring the Pseudowire Class, on page 25.

How AToM Transports Layer 2 Packets Using Commands Associated with L2VPN Protocol-Based Feature

AToM encapsulates Layer 2 frames at the ingress PE and sends them to a corresponding PE at the other end of a pseudowire, which is a connection between the two PE routers. The egress PE removes the encapsulation and sends out the Layer 2 frame.

The successful transmission of the Layer 2 frames between PE routers is due to the configuration of the PE routers. You set up the connection, called a pseudowire, between the routers. You specify the following information on each PE router:

- The type of Layer 2 data that will be transported across the pseudowire, such as Ethernet, Frame Relay, or ATM
- The IP address of the loopback interface of the peer PE router, which enables the PE routers to communicate
- A unique combination of peer PE IP address and VC ID that identifies the pseudowire

The following example shows the basic configuration steps on a PE router that enable the transport of Layer 2 packets. Each transport type has slightly different steps.

Step 1 defines the interface or subinterface on the PE router:

```
Router# interface
interface-type interface-number
```

```
Router(config)# interface gi 0/1/0
```

Step 2 configures an ethernet service instance on an interface and enters service instance configuration mode:

```
Router(config-if)#service instance number ethernet WORD
Router(config-if)# service instance 393 ethernet ethernet1
```

Step 3 specifies the encapsulation type for the interface, such as dot1q:

```
Router(config-if)# encapsulation
encapsulation-type
```

```
Router(config-if-srv)# encapsulation dot1q 393
```

Step 3 does the following:

- Makes a connection to the peer PE router by specifying the LDP router ID of the peer PE router.
- Specifies a 32-bit unique identifier, called the VC ID, which is shared between the two PE routers.

The combination of the peer router ID and the VC ID must be unique on the router. Two circuits cannot use the same combination of peer router ID and VC ID.

- Specifies the tunneling method used to encapsulate data in the pseudowire. AToM uses MPLS as the tunneling method.

```
Router(config)# interface pseudowire 100
Router(config-if)# encapsulation mpls
Router(config-if)# neighbor 10.0.0.1 123
Router(config-if)# exit
!
Router(config)# l2vpn xconnect context A
Router(config-xconnect)# member pseudowire 100
Router(config-xconnect)# member gigabitethernet0/0/0.1
Router (config-xconnect)# member gigabitethernet0/1/0 service instance 393

Router(config-xconnect)# exit
```

As an alternative, you can set up a pseudowire class to specify the tunneling method and other characteristics. For more information, see the Configuring the Pseudowire Class, on page 25.

Benefits of AToM

The following list explains some of the benefits of enabling Layer 2 packets to be sent in the MPLS network:

- The AToM product set accommodates many types of Layer 2 packets, including Ethernet and Frame Relay, across multiple Cisco router platforms. This enables the service provider to transport all types of traffic over the backbone and accommodate all types of customers.
- AToM adheres to the standards developed for transporting Layer 2 packets over MPLS. This benefits the service provider that wants to incorporate industry-standard methodologies in the network. Other Layer 2 solutions are proprietary, which can limit the service provider's ability to expand the network and can force the service provider to use only one vendor's equipment.
- Upgrading to AToM is transparent to the customer. Because the service provider network is separate from the customer network, the service provider can upgrade to AToM without disruption of service to the customer. The customers assume that they are using a traditional Layer 2 backbone.

MPLS Traffic Engineering Fast Reroute

Note For the supported combinations of MPLS TE FRR on Cisco RSP3 Module, see the *MPLS Traffic Engineering Path Link and Node Protection Configuration Guide*.

AToM can use MPLS traffic engineering (TE) tunnels with fast reroute (FRR) support. AToM VCs can be rerouted around a failed link or node at the same time as MPLS and IP prefixes.

Enabling fast reroute on AToM does not require any special commands; you can use standard fast reroute commands. At the ingress PE, an AToM tunnel is protected by fast reroute when it is routed to an FRR-protected TE tunnel. Both link and node protection are supported for AToM VCs at the ingress PE.

In the following example, the primary link is disabled, which causes the backup tunnel (Tunnel 1) to become the primary path. The output in boldface font shows the status of the tunnel:

```
Router# execute-on slot 3 debug mpls l2transport fast-reroute
========= Line Card (Slot 3) =========
AToM fast reroute debugging is on
SLOT 3:Sep 16 17:58:56.346: AToM SMGR: Processing TFIB FRR event for 10.4.0.1
SLOT 3:Sep 16 17:58:56.346: AToM SMGR: Finished processing TFIB FRR event for 10.4.0.1
SLOT 3:Sep 16 17:58:56.346: AToM SMGR: Processing TFIB FRR event for Tunnel41
SLOT 3:Sep 16 17:58:56.346: AToM SMGR: Finished processing TFIB FRR event for Tunnel41
Sep 16 17:58:58.342: %LINK-3-UPDOWN: Interface POS0/0/0, changed state to down
Sep 16 17:58:58.342: %OSPF-5-ADJCHG: Process 1, Nbr 10.0.0.1 on POS0/0 from FULL to DOWN,
Neighbor Down: Interface down or detached
Sep 16 17:58:59.342: %LINEPROTO-5-UPDOWN: Line protocol on Interface POS0/0/0, changed state
 to down
```

Maximum Transmission Unit Guidelines for Estimating Packet Size

The following calculation helps you determine the size of the packets traveling through the core network. You set the maximum transmission unit (MTU) on the core-facing interfaces of the P and PE routers to accommodate packets of this size. The MTU should be greater than or equal to the total bytes of the items in the following equation:

```
Core MTU >= (Edge MTU + Transport header + AToM header + (MPLS label stack * MPLS label
size))
```

The following sections describe the variables used in the equation.

Edge MTU

The edge MTU is the MTU for the customer-facing interfaces.

Transport Header

The Transport header depends on the transport type. The table below lists the specific sizes of the headers.

Table 2: Header Size of Packets

Transport Type	Packet Size
AAL5	0-32 bytes
Ethernet VLAN	18 bytes
Ethernet Port	14 bytes
Frame Relay DLCI	2 bytes for Cisco encapsulation, 8 bytes for Internet Engineering Task Force (IETF) encapsulation
HDLC	4 bytes
PPP	4 bytes

AToM Header

The AToM header is 4 bytes (control word). The control word is optional for Ethernet, PPP, HDLC, and cell relay transport types. The control word is required for Frame Relay and ATM AAL5 transport types.

MPLS Label Stack

The MPLS label stack size depends on the configuration of the core MPLS network:

- AToM uses one MPLS label to identify the AToM VCs (VC label). Therefore, the minimum MPLS label stack is one for directly connected AToM PEs, which are PE routers that do not have a P router between them.
- If LDP is used in the MPLS network, the label stack size is two (the LDP label and the VC label).
- If a TE tunnel instead of LDP is used between PE routers in the MPLS network, the label stack size is two (the TE label and the VC label).
- If a TE tunnel and LDP are used in the MPLS network (for example, a TE tunnel between P routers or between P and PE routers, with LDP on the tunnel), the label stack is three (TE label, LDP label, VC label).
- If you use MPLS fast reroute in the MPLS network, you add a label to the stack. The maximum MPLS label stack in this case is four (FRR label, TE label, LDP label, VC label).
- If AToM is used by the customer carrier in an MPLS VPN Carrier Supporting Carrier environment, you add a label to the stack. The maximum MPLS label stack in the provider carrier network is five (FRR label, TE label, LDP label, VPN label, VC label)four (FRR label, TE label, LDP label, VC label).
- BGP PIC Edge with EoMPLS using BGP label Unicast (RFC 3107) requires the **bgp mpls-local-label** command to be explicitly enabled under the Router BGP process. This limitation is applicable only on the Cisco RSP3 module.
- BGP PIC Edge with EoMPLS/VPLS/EVPN using BGP label Unicast (RFC 3107) requires the **bgp mpls-local-label** command to be explicitly enabled under the Router BGP process to forward the data plane traffic. This limitation is applicable on the Cisco ASR 900 RSP3 module and Cisco ASR 907
- If an AToM tunnel spans different service providers that exchange MPLS labels using IPv4 Border Gateway Protocol (BGP) (RFC 3107), you add a label to the stack. The maximum MPLS label stack is five (FRR label, TE label, LDP label, VPN label, VC label)four (FRR label, TE label, LDP label, VC label)
- TE-FRR with BGP labels for layer 2 and layer 3 VPNs must terminate on the BGP gateway because of the four-label limitation.

Other circumstances can increase the MPLS label stack size. Therefore, analyze the complete data path between the AToM tunnel endpoints and determine the maximum MPLS label stack size for your network. Then multiply the label stack size by the size of the MPLS label.

Hot standby pseudowire (HSPW) convergence without pseudowire grouping increments linearly, with a thousand virtual circuits taking 54 seconds of convergence time. This is applicable only on the Cisco RSP3 Module.

Estimating Packet Size Example

The estimated packet size in the following example is 1526 bytes, based on the following assumptions:

- The edge MTU is 1500 bytes.
- The transport type is Ethernet VLAN, which designates 18 bytes for the transport header.
- The AToM header is 0, because the control word is not used.
- The MPLS label stack is 2, because LDP is used. The MPLS label is 4 bytes.

```
Edge MTU + Transport header + AToM header + (MPLS label stack * MPLS label) = Core MTU
1500     + 18                     + 0      + (2                * 4          ) = 1526
```

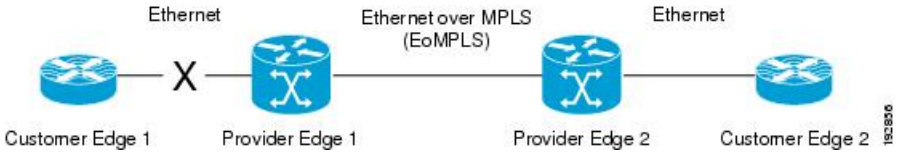
You must configure the P and PE routers in the core to accept packets of 1526 bytes.

Any Transport over MPLS (AToM) Remote Ethernet Port Shutdown

This Cisco IOS XE feature allows a service provider edge (PE) router on the local end of an Ethernet over MPLS (EoMPLS) pseudowire to detect a remote link failure and cause the shutdown of the Ethernet port on the local customer edge (CE) router. Because the Ethernet port on the local CE router is shut down, the router does not lose data by continuously sending traffic to the failed remote link. This is beneficial if the link is configured as a static IP route.

The figure below illustrates a condition in an EoMPLS WAN, with a down Layer 2 tunnel link between a CE router (Customer Edge 1) and the PE router (Provider Edge 1). A CE router on the far side of the Layer 2 tunnel (Customer Edge 2), continues to forward traffic to Customer Edge 1 through the L2 tunnel.

Figure 1: Remote Link Outage in EoMPLS WAN



Previous to this feature, the Provider Edge 2 router could not detect a failed remote link. Traffic forwarded from Customer Edge 2 to Customer Edge 1 would be lost until routing or spanning tree protocols detected the down remote link. If the link was configured with static routing, the remote link outage would be even more difficult to detect.

With this feature, the Provider Edge 2 router detects the remote link failure and causes a shutdown of the local Customer Edge 2 Ethernet port. When the remote L2 tunnel link is restored, the local interface is automatically restored as well. The possibility of data loss is thus diminished.

With reference to the figure above, the Remote Ethernet Shutdown sequence is generally described as follows:

1. The remote link between Customer Edge 1 and Provider Edge 1 fails.
2. Provider Edge 2 detects the remote link failure and disables the transmit laser on the line card interface connected to Customer Edge 2.
3. An RX_LOS error alarm is received by Customer Edge 2 causing Customer Edge 2 to bring down the interface.
4. Provider Edge 2 maintains its interface with Customer Edge 2 in an up state.
5. When the remote link and EoMPLS connection is restored, the Provider Edge 2 router enables the transmit laser.

6. The Customer Edge 2 router brings up its downed interface.

This feature is enabled by default for Ethernet over MPLS (EoMPLS). You can also enable this feature by using the **remote link failure notification** command in xconnect configuration mode as shown in the following example:

```
pseudowire-class eompls
 encapsulation mpls
!
interface GigabitEthernet1/0/0
 xconnect 10.13.13.13 1 pw-class eompls
  remote link failure notification
!
```

This feature can be disabled using the **no remote link failure notification** command in xconnect configuration mode. Use the **show ip interface brief** privileged EXEC command to display the status of all remote L2 tunnel links. Use the **show interface** privileged EXEC command to show the status of the L2 tunnel on a specific interface.

Note The **no remote link failure notification** command will not give notification to clients for remote attachment circuit status down.

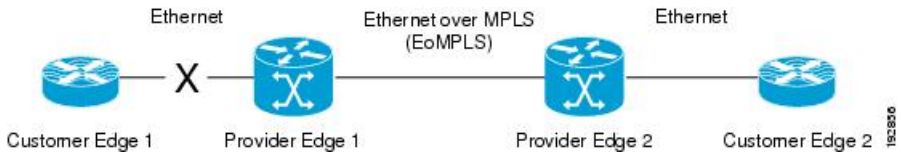
Note Remote Ethernet Port Shutdown is supported only on EFP with encapsulation default.

Any Transport over MPLS (AToM) Remote Ethernet Port Shutdown Using Commands Associated with L2VPN Protocol-Based Feature

This Cisco IOS XE feature allows a service provider edge (PE) router on the local end of an Ethernet over MPLS (EoMPLS) pseudowire to detect a remote link failure and cause the shutdown of the Ethernet port on the local customer edge (CE) router. Because the Ethernet port on the local CE router is shut down, the router does not lose data by continuously sending traffic to the failed remote link. This is beneficial if the link is configured as a static IP route.

The figure below illustrates a condition in an EoMPLS WAN, with a down Layer 2 tunnel link between a CE router (Customer Edge 1) and the PE router (Provider Edge 1). A CE router on the far side of the Layer 2 tunnel (Customer Edge 2), continues to forward traffic to Customer Edge 1 through the L2 tunnel.

Figure 2: Remote Link Outage in EoMPLS WAN



Previous to this feature, the Provider Edge 2 router could not detect a failed remote link. Traffic forwarded from Customer Edge 2 to Customer Edge 1 would be lost until routing or spanning tree protocols detected the down remote link. If the link was configured with static routing, the remote link outage would be even more difficult to detect.

With this feature, the Provider Edge 2 router detects the remote link failure and causes a shutdown of the local Customer Edge 2 Ethernet port. When the remote L2 tunnel link is restored, the local interface is automatically restored as well. The possibility of data loss is thus diminished.

With reference to the figure above, the Remote Ethernet Shutdown sequence is generally described as follows:

1. The remote link between Customer Edge 1 and Provider Edge 1 fails.
2. Provider Edge 2 detects the remote link failure and disables the transmit laser on the line card interface connected to Customer Edge 2.
3. An RX_LOS error alarm is received by Customer Edge 2 causing Customer Edge 2 to bring down the interface.
4. Provider Edge 2 maintains its interface with Customer Edge 2 in an up state.
5. When the remote link and EoMPLS connection is restored, the Provider Edge 2 router enables the transmit laser.
6. The Customer Edge 2 router brings up its downed interface.

This feature is enabled by default for Ethernet over MPLS (EoMPLS). You can also enable this feature by using the **remote link failure notification** command in xconnect configuration mode as shown in the following example:

```
template type pseudowire eompls
 encapsulation mpls
!
interface Pseudowire 100
 source template type pseudowire test
 neighbor 10.13.13.13 1
interface GigabitEthernet1/0/0
 service instance 300 ethernet
 encapsulation default
 xconnect 10.1.1.1 1 encapsulation mpls
 remote link failure notification
l2vpn xconnect context con1
 member GigabitEthernet1/0/0 service-instance 300
 member Pseudowire 100
!

l2vpn xconnect context con1
member GigabitEthernet1/0/0 service-instance 300
member Pseudowire 100
remote link failure notification
```

This feature can be disabled using the **no remote link failure notification** command in xconnect configuration mode. Use the **show ip interface brief** privileged EXEC command to display the status of all remote L2 tunnel links. Use the **show interface** privileged EXEC command to show the status of the L2 tunnel on a specific interface.

Note The **no remote link failure notification** command will not give notification to clients for remote attachment circuit status down.

Flow-Aware Transport (FAT) Load Balancing

Note The FAT-PW feature is supported only in the RSP3 module and only with the new CLI.

The Flow-Aware Transport of MPLS Pseudowires feature enables load balancing of packets within the same pseudowire by further classifying the packets into different flows by adding a flow label at the bottom of the MPLS label stack.

Equal Cost Multi-Path

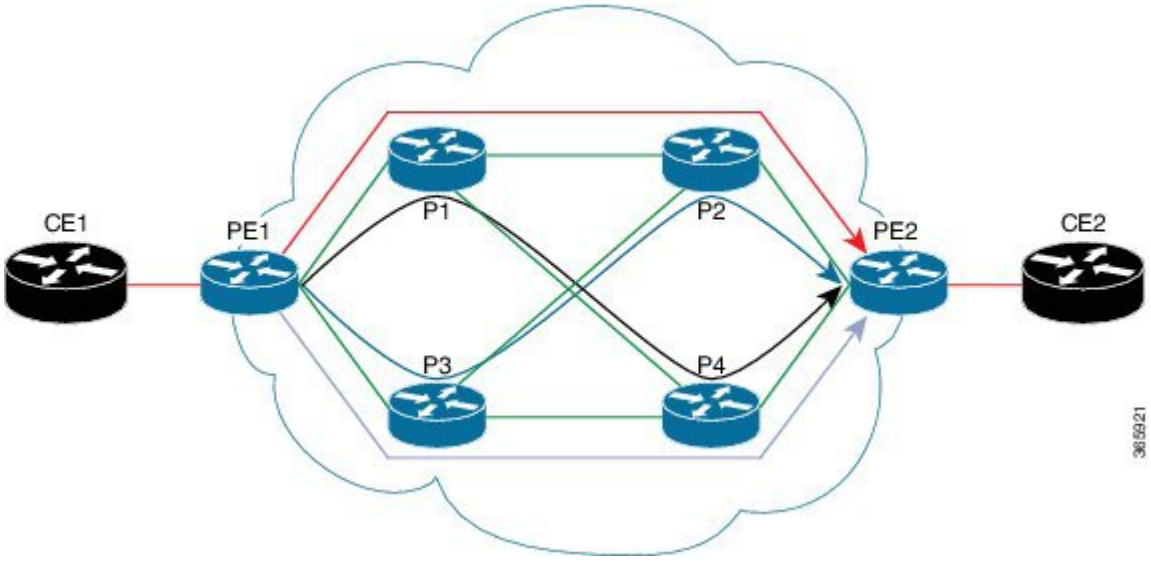
The Flow-Aware Transport Pseudowire (FAT-PW) is used to load-balance traffic in the core when Equal Cost Multiple Paths (ECMP) exist. The existing Load Balance technique does the load balance among multiple pseudowires by choosing different ECMP paths, based on the Virtual Circuit (VC) Label. This does not suffice the load balance of traffic within a pseudowire.

A flow label is a unique identifier to distinguish a flow within the pseudowire and is generated based on source and destination MAC address along with source and destination IP address. The flow label has EOS (End of Label Stack) bit SET and inserted before the VC label and after the control word, if necessary. Calculation and pushing of the flow label is done by an ingress PE, enabled by FAT-PW configuration. Egress PE discards the flow label and no decisions are taken based on that label.

All core routers do a load balance based on the bottom-most label, which is a flow-label in FAT-PW. Hence you get the advantage of distributing flows over ECMP paths.

The figure below shows the various paths through which the data can be transmitted in an ECMP.

Figure 3: Equal Cost Multi-Path



- Without any load-balancing, the pseudowire can use any one path of the four options, for example consider the red path (PE1 > P1 > P2 > PE2)
- If PE1 is able to do load-balancing, then both PE1 and PE2 can be utilized, for example consider the red and gray paths (PE1 > P3 > P4 > PE2)

- With flow labels inserted on PE1, all paths can be utilized, for example red, black, blue, and gray paths

How to Configure Any Transport over MPLS

This section explains how to perform a basic AToM configuration and includes the following procedures:

Configuring the Pseudowire Class

Note In simple configurations, this task is optional. You need not specify a pseudowire class if you specify the tunneling method as part of the **xconnect** command.

- You must specify the **encapsulation mpls** command as part of the pseudowire class or as part of the **xconnect** command for the AToM VCs to work properly. If you omit the **encapsulation mpls** command as part of the **xconnect** command, you receive the following error:

```
% Incomplete command.
```

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **pseudowire-class** *name*

Example:

```
Router(config)# pseudowire-class atom
```

Establishes a pseudowire class with a name that you specify and enters pseudowire class configuration mode.

Step 4 **encapsulation mpls**

Example:

```
Router(config-pw)# encapsulation mpls
```

Specifies the tunneling encapsulation.

Configuring the Pseudowire Class Using Commands Associated with L2VPN Protocol-Based Feature

Note In simple configurations, this task is optional. You need not specify a pseudowire class if you specify the tunneling method as part of the **l2vpn xconnect context** command.

- You must specify the **encapsulation mpls** command as part of the pseudowire class or as part of the **l2vpn xconnect context** command for the AToM VCs to work properly. If you omit the **encapsulation mpls** command as part of the **l2vpn xconnect context**command, you receive the following error:

```
% Incomplete command.
```

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **interface pseudowire** *name*

Example:

```
Router(config)# interface pseudowire atom
```

Establishes an interface pseudowire with a name that you specify and enters pseudowire class configuration mode.

Step 4 **encapsulation mpls**

Example:

```
Router(config-pw-class)# encapsulation mpls
```

Specifies the tunneling encapsulation.

Step 5 **neighbor** *peer-address vcid-value*

Example:

```
Router(config-pw-class)#  neighbor 33.33.33.33 1
```

Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.

Changing the Encapsulation Type and Removing a Pseudowire

Once you specify the **encapsulation mpls** command, you cannot remove it using the **no encapsulation mpls** command.

Nor can you change the command's setting using the **encapsulation l2tpv3** command.

Those methods result in the following error message:

```
Encapsulation changes are not allowed on an existing pw-class.
```

To remove the **encapsulation mpls** command, you must delete the pseudowire with the **no pseudowire-class** command.

To change the type of encapsulation, remove the pseudowire using the **no pseudowire-class** command and reconfigure the pseudowire to specify the new encapsulation type.

Changing the Encapsulation Type and Removing a Pseudowire Using Commands Associated with the L2VPN Protocol-Based Feature

Once you specify the **encapsulation mpls** command, you cannot remove it using the **no encapsulation mpls** command.

Nor can you change the command's setting using the **encapsulation l2tpv3** command.

Those methods result in the following error message:

```
Encapsulation changes are not allowed on an existing pw-class.
% Cannot remove encapsulation on existing pseudowire
```

To remove the **encapsulation mpls** command, you must delete the pseudowire with the **no interface pseudowire** command.

To change the type of encapsulation, remove the pseudowire using the **no template type pseudowire** command and reconfigure the pseudowire to specify the new encapsulation type.

Configuring ATM AAL5 over MPLS

Configuring ATM AAL5 over MPLS on PVCs

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type slot / subslot / port* [**.** *subinterface*]

Example:

```
Router(config)# interface atm1/0/0
```

Specifies the interface type and enters interface configuration mode.

Step 4 **pvc** [*name*] *vpi / vci* **l2transport**

Example:

```
Router(config-if)# pvc 1/200 l2transport
```

Creates or assigns a name to an ATM PVC and enters L2transport PVC configuration mode.

- The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.

Step 5 **encapsulation aal5**

Example:

```
Router(config-if-atm-l2trans-pvc)# encapsulation aal5
```

Specifies ATM AAL5 encapsulation for the PVC. Make sure you specify the same encapsulation type on the PE and customer edge (CE) routers.

Step 6 **xconnect** *peer-router-id vcid* **encapsulation mpls**

Example:

```
Router(config-if-atm-l2trans-pvc)# xconnect 10.13.13.13 100 encapsulation mpls
```

Binds the attachment circuit to a pseudowire VC.

Step 7 **end**

Example:

```
Router(config-if-atm-l2trans-pvc)# end
```

Exits to privileged EXEC mode.

Step 8 **show mpls l2transport vc**

Example:

```
Router# show mpls l2transport vc
```

Displays output that shows ATM AAL5 over MPLS is configured on a PVC.

Examples

The following is sample output from the **show mpls l2transport vc** command that shows that ATM AAL5 over MPLS is configured on a PVC:

```
Router# show mpls l2transport vc
Local intf   Local circuit          Dest address      VC ID      Status
---------    -------------          ------------      -----      ------
ATM1/0       ATM AAL5 1/100         10.4.4.4           100        UP
```

Configuring ATM AAL5 over MPLS on PVCs using the commands associated with the L2VPN Protocol-Based CLIs feature

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type slot* / *subslot* / *port*[**.** *subinterface*]

Example:

```
Device(config)# interface atm1/0/0
```

Specifies the interface type and enters interface configuration mode.

Step 4 **pvc** [*name*] *vpi* / *vci* **l2transport**

Example:

```
Device(config-if)# pvc 1/200 l2transport
```

Creates or assigns a name to an ATM PVC and enters L2transport PVC configuration mode.

- The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.

Step 5 **encapsulation aal5**

Example:

```
Device(config-if-atm-l2trans-pvc)# encapsulation aal5
```

Specifies ATM AAL5 encapsulation for the PVC. Make sure you specify the same encapsulation type on the PE and customer edge (CE) routers.

Step 6 **end**

Example:

```
Device(config-if-atm-l2trans-pvc)# end
```

Exits to privileged EXEC mode.

Step 7 **interface pseudowire** *number*

Example:

```
Device(config)# interface pseudowire 100
```

Specifies the pseudowire interface and enters interface configuration mode.

Step 8 **encapsulation mpls**

Example:

```
Device(config-if)# encapsulation mpls
```

Specifies that Multiprotocol Label Switching (MPLS) is used as the data encapsulation method.

Step 9 **neighbor** *peer-address vcid-value*

Example:

```
Device(config-if)# neighbor 10.13.13.13 100
```

Specifies the peer IP address and virtual circuit (VC) ID value of the Layer 2 VPN (L2VPN) pseudowire.

Step 10 **exit**

Example:

```
Device(config-if)# exit
```

Exits interface configuration mode.

Step 11 **l2vpn xconnect context** *context-name*

Example:

```
Device(config)# l2vpn xconnect context con1
```

Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.

Step 12 **member pseudowire** *interface-number*

Example:

```
Device(config-xconnect)# member pseudowire 100
```

Specifies a member pseudowire to form a Layer 2 VPN (L2VPN) cross connect.

Step 13 **member atm** *interface-number* **pvc** *vpi* / *vci*

Example:

```
Device(config-xconnect)# member atm 100 pvc 1/200
```

Specifies the location of the ATM member interface.

Step 14 **end**

Example:

```
Device(config-xconnect)# end
```

Exits to privileged EXEC mode.

Step 15 **show l2vpn atom vc**

Example:

```
Device# show l2vpn atom vc
```

Displays output that shows ATM AAL5 over MPLS is configured on a PVC.

Examples

The following is sample output from the **show l2vpn atom vc** command that shows that ATM AAL5 over MPLS is configured on a PVC:

```
Device# show l2vpn atom vc
Local intf   Local circuit          Dest address      VC ID      Status
---------    -------------          ------------      -----      ------
ATM1/0       ATM AAL5 1/100         10.4.4.4           100        UP
```

Configuring ATM AAL5 over MPLS in VC Class Configuration Mode

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **vc-class atm** *vc-class-name*

Example:

```
Router(config)# vc-class atm aal5class
```

Creates a VC class and enters VC class configuration mode.

Step 4 **encapsulation** *layer-type*

Example:

```
Router(config-vc-class)# encapsulation aal5
```

Configures the AAL and encapsulation type.

Step 5 **exit**

Example:

```
Router(config-vc-class)# exit
```

Exits VC class configuration mode.

Step 6 **interface** *type slot* **/** *subslot* **/** *port* [**.** *subinterface*]

Example:

```
Router(config)# interface atm1/0/0
```

Specifies the interface type enters interface configuration mode.

Step 7 **class-int** *vc-class-name*

Example:

```
Router(config-if)# class-int aal5class
```

Applies a VC class to the ATM main interface or subinterface.

Note You can also apply a VC class to a PVC.

Step 8 **pvc** [*name*] *vpi* / *vci* **l2transport**

Example:

```
Router(config-if)# pvc 1/200 l2transport
```

Creates or assigns a name to an ATM PVC and enters L2transport PVC configuration mode.

- The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.

Step 9 **xconnect** *peer-router-id vcid* **encapsulation mpls**

Example:

```
Router(config-if-atm-l2trans-pvc)# xconnect 10.13.13.13 100 encapsulation mpls
```

Binds the attachment circuit to a pseudowire VC.

Step 10 **end**

Example:

```
Router(config-if-atm-l2trans-pvc)# end
```

Exits to privileged EXEC mode.

Step 11 **show atm class-links**

Example:

```
Router# show atm class-links
```

Displays the type of encapsulation and that the VC class was applied to an interface.

Examples

In the following example, the command output from the **show atm class-links** command verifies that ATM AAL5 over MPLS is configured as part of a VC class. The command output shows the type of encapsulation and that the VC class was applied to an interface.

```
Router# show atm class-links 1/100
Displaying vc-class inheritance for ATM1/0/0.0, vc 1/100:
no broadcast - Not configured - using default
encapsulation aal5 - VC-class configured on main interface
```

Configuring ATM AAL5 over MPLS in VC Class Configuration Mode using the commands associated with the L2VPN Protocol-Based CLIs feature

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **vc-class atm** *vc-class-name*

Example:

```
Router(config)# vc-class atm aal5class
```

Creates a VC class and enters VC class configuration mode.

Step 4 **encapsulation** *layer-type*

Example:

```
Router(config-vc-class)# encapsulation aal5
```

Configures the AAL and encapsulation type.

Step 5 **exit**

Example:

```
Router(config-vc-class)# exit
```

Exits VC class configuration mode.

Step 6 **interface** *type slot* **/** *subslot* **/** *port* [**.** *subinterface*]

Example:

```
Router(config)# interface atm1/0/0
```

Specifies the interface type enters interface configuration mode.

Step 7 **class-int** *vc-class-name*

Example:

```
Router(config-if)# class-int aal5class
```

Applies a VC class to the ATM main interface or subinterface.

Note You can also apply a VC class to a PVC.

Step 8 **pvc** [*name*] *vpi* / *vci* **l2transport**

Example:

```
Router(config-if)# pvc 1/200 l2transport
```

Creates or assigns a name to an ATM PVC and enters L2transport PVC configuration mode.

- The **l2transport** keyword indicates that the PVC is a switched PVC instead of a terminated PVC.

Step 9 **exit**

Example:

```
Router(config-if)# exit
```

Exits interface configuration mode.

Step 10 **interface pseudowire** *number*

Example:

```
Router(config)# interface pseudowire 100
```

Specifies the pseudowire interface and enters interface configuration mode.

Step 11 **encapsulation mpls**

Example:

```
Router(config-if)# encapsulation mpls
```

Specifies that Multiprotocol Label Switching (MPLS) is used as the data encapsulation method.

Step 12 **neighbor** *peer-address vcid-value*

Example:

```
Router(config-if)# neighbor 10.0.0.1 123
```

Specifies the peer IP address and virtual circuit (VC) ID value of the Layer 2 VPN (L2VPN) pseudowire.

Step 13 **exit**

Example:

```
Router(config-if)# exit
```

Exits interface configuration mode.

Step 14 **l2vpn xconnect context** *context-name*

Example:

```
Router(config)# l2vpn xconnect context con1
```

Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.

Step 15 **member pseudowire** *interface-number*

Example:

```
Router(config-xconnect)# member pseudowire 100
```

Specifies a member pseudowire to form a Layer 2 VPN (L2VPN) cross connect.

Step 16 **member atm** *interface-number*

Example:

```
Device(config-xconnect)# member atm 100
```

Specifies the location of the ATM member interface.

Step 17 **end**

Example:

```
Router(config-if-atm-l2trans-pvc)# end
```

Exits to privileged EXEC mode.

Step 18 **show atm class-links**

Example:

```
Router# show atm class-links
```

Displays the type of encapsulation and that the VC class was applied to an interface.

Examples

In the following example, the command output from the **show atm class-links** command verifies that ATM AAL5 over MPLS is configured as part of a VC class. The command output shows the type of encapsulation and that the VC class was applied to an interface.

```
Router# show atm class-links 1/100
Displaying vc-class inheritance for ATM1/0/0.0, vc 1/100:
no broadcast - Not configured - using default
encapsulation aal5 - VC-class configured on main interface
```

Configuring Ethernet over MPLS

Configuring Ethernet over MPLS with VLAN ID Rewrite

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **interface gigabitethernet** *slot* **/** *subslot* **/** *port* [**.** *subinterface*]

Example:

```
Router(config)# interface gigabitethernet4/0/0.1
```

```
Router(config)# interface GigabitEthernet0/2/4
```

Specifies the Gigabit Ethernet subinterface and enters subinterface configuration mode.

Step 4 **no ip address**

Example:

```
Router(config-if)# no ip address
```

Specifies that there is no IP address assigned to the interface.

Step 5 **negotiation auto**

Example:

```
Router(config-if)# negotiation auto
```

Enables the auto negotiation protocol.

Step 6 **service instance** *id* **ethernet**

Example:

```
Router(config-if)# service instance 100 ethernet
```

Configures an ethernet service instance on an interface and enters service instance configuration mode.

Step 7 **encapsulation dot1q** *vlan-id*

Example:

```
Router(config-subif)# encapsulation dot1q 100
```

Enables the subinterface to accept 802.1Q VLAN packets.

Step 8 **xconnect** *peer-router-id vcid* **encapsulation mpls**

Example:

```
Router(config-subif)# xconnect 10.0.0.1 123 encapsulation mpls
```

Binds the attachment circuit to a pseudowire VC and enters xconnect configuration mode.

Step 9 **remote circuit id** *remote-vlan-id*

Example:

```
Router(config-subif-xconn)# remote circuit id 101
```

(Optional) Enables you to use VLAN interfaces with different VLAN IDs at both ends of the tunnel.

Step 10 **end**

Example:

```
Router(config-subif-xconn)# end
```

Exits to privileged EXEC mode.

Step 11 **show controllers eompls forwarding-table**

Example:

```
Router# show controllers eompls forwarding-table
```

Displays information about VLAN ID rewrite.

Examples

On PE1

On PE2

The following sample output from the **show controllers eompls forwarding-table** command shows VLAN ID rewrite configured on a router with an engine 2 3-port Gigabit Ethernet line card. In this example, the output in boldface font shows the VLAN ID rewrite information.

```
Router# execute slot 0 show controllers eompls forwarding-table 0 2
Port # 0, VLAN-ID # 2, Table-index 2
EoMPLS configured: 1
tag_rew_ptr             = D001BB58
Leaf entry?     = 1
FCR index       = 20
```

```
          **tagrew_psa_addr    = 0006ED60
          **tagrew_vir_addr    = 7006ED60
          **tagrew_phy_addr    = F006ED60
        [0-7] loq 8800 mtu 4458  oq 4000 ai 3 oi 04019110 (encaps size 4)
        cw-size 4 vlanid-rew 3
        gather A30 (bufhdr size 32 EoMPLS (Control Word) Imposition profile 81)
        2 tag: 18 18
        counters 1182, 10 reported 1182, 10.
    Local OutputQ (Unicast):    Slot:2  Port:0  RED queue:0  COS queue:0
    Output Q (Unicast):         Port:0          RED queue:0  COS queue:0

Router# execute slot 0 show controllers eompls forwarding-table 0 3
Port # 0, VLAN-ID # 3, Table-index 3
EoMPLS configured: 1
tag_rew_ptr             = D0027B90
Leaf entry?     = 1
FCR index       = 20
          **tagrew_psa_addr    = 0009EE40
          **tagrew_vir_addr    = 7009EE40
          **tagrew_phy_addr    = F009EE40
        [0-7] loq 9400 mtu 4458  oq 4000 ai 8 oi 84000002 (encaps size 4)
        cw-size 4 vlanid-rew 2
        gather A30 (bufhdr size 32 EoMPLS (Control Word) Imposition profile 81)
        2 tag: 17 18
        counters 1182, 10 reported 1182, 10.
    Local OutputQ (Unicast):    Slot:5  Port:0  RED queue:0  COS queue:0
    Output Q (Unicast):         Port:0          RED queue:0  COS queue:0
```

Configuring Ethernet over MPLS with VLAN ID Rewrite Using Commands Associated with the L2VPN Protocol-Based Feature

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **interface gigabitethernet** *slot* / *subslot* / *port* [.*subinterface*]

Example:

```
Router(config)# interface gigabitethernet4/0/0.1
```

Specifies the Gigabit Ethernet subinterface and enters subinterface configuration mode.

Step 4 **interface gigabitethernet** *slot* / *subslot* / *port*

Example:

```
Router(config)# interface gigabitethernet4/0/0
```

Specifies the Gigabit Ethernet subinterface and enters subinterface configuration mode.

Step 5 **service instance** *number* **ethernet** *number*

Example:

```
Router(config-if)#service instance 393 ethernet
```

Step 6 **encapsulation dot1q** *vlan-id*

Example:

```
Router(config-subif)# encapsulation dot1q 100
```

Enables the subinterface to accept 802.1Q VLAN packets.

Step 7 **end**

Example:

```
Router(config-subif)# end
```

Exits to privileged EXEC mode.

Step 8 **interface pseudowire** *number*

Example:

```
Router(config)# interface pseudowire 100
```

Specifies the pseudowire interface and enters interface configuration mode.

Step 9 **encapsulation mpls**

Example:

```
Router(config-if)# encapsulation mpls
```

Specifies that Multiprotocol Label Switching (MPLS) is used as the data encapsulation method.

Step 10 **neighbor** *peer-address vcid-value*

Example:

```
Router(config-if)# neighbor 10.0.0.1 123
```

Specifies the peer IP address and virtual circuit (VC) ID value of the Layer 2 VPN (L2VPN) pseudowire.

Step 11 **exit**

Example:

```
Router(config-if)# exit
```

Exits interface configuration mode.

Step 12 **l2vpn xconnect context** *context-name*

Example:

```
Router(config)# l2vpn xconnect context con1
```

Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.

Step 13 **member pseudowire** *interface-number*

Example:

```
Router(config-xconnect)# member pseudowire 100
```

Specifies a member pseudowire to form a Layer 2 VPN (L2VPN) cross connect.

Step 14 **member gigabitethernet** *interface-number*

Example:

```
Router(config-xconnect)# member GigabitEthernet0/0/0.1

Router(config-xconnect)# member gigabitethernet4/0/0 service-instance 393
```

Specifies the location of the Gigabit Ethernet member interface.

Step 15 **remote circuit id** *remote-vlan-id*

Example:

```
Router(config-xconnect)# remote circuit id 101
```

(Optional) Enables you to use VLAN interfaces with different VLAN IDs at both ends of the tunnel.

Step 16 **end**

Example:

```
Router(config-xconnect)# end
```

Exits to privileged EXEC mode.

Step 17 **show controllers eompls forwarding-table**

Example:

```
Router# show controllers eompls forwarding-table
```

Displays information about VLAN ID rewrite.

ExamplesExample

On PE1

On PE2

```
RSP3-RT1#show ethernet service instance id HYPERLINK "tel:1002"1002 interface gi 0/1/0 det
Service Instance ID: HYPERLINK "tel:1002"1002
Service Instance Type: Static
Associated Interface: GigabitEthernet0/1/0
Associated EVC:
L2protocol drop
CE-Vlans:
Encapsulation: dot1q HYPERLINK "tel:1002"1002 vlan protocol type 0xHYPERLINK "tel:8100"8100
Rewrite: ingress tag pop 1 symmetric
Interface Dot1q Tunnel Ethertype: 0xHYPERLINK "tel:8100"8100
State: Up
EFP Statistics:
Pkts In   Bytes In   Pkts Out  Bytes Out
0          0          0          0

RSP3-RT1#
```

The following sample output from the **show controllers eompls forwarding-table** command shows VLAN ID rewrite configured on a router with an engine 2 3-port Gigabit Ethernet line card. In this example, the output in boldface font shows the VLAN ID rewrite information.

```
Router# execute slot 0 show controllers eompls forwarding-table 0 2
Port # 0, VLAN-ID # 2, Table-index 2
EoMPLS configured: 1
tag_rew_ptr             = D001BB58
Leaf entry?     = 1
FCR index       = 20
           **tagrew_psa_addr    = 0006ED60
           **tagrew_vir_addr    = 7006ED60
           **tagrew_phy_addr    = F006ED60
        [0-7] loq 8800 mtu 4458  oq 4000 ai 3 oi 04019110 (encaps size 4)
        cw-size 4 vlanid-rew 3
        gather A30 (bufhdr size 32 EoMPLS (Control Word) Imposition profile 81)
        2 tag: 18 18
        counters 1182, 10 reported 1182, 10.
    Local OutputQ (Unicast):    Slot:2  Port:0  RED queue:0  COS queue:0
    Output Q (Unicast):         Port:0          RED queue:0  COS queue:0

Router# execute slot 0 show controllers eompls forwarding-table 0 3
Port # 0, VLAN-ID # 3, Table-index 3
EoMPLS configured: 1
tag_rew_ptr             = D0027B90
Leaf entry?     = 1
FCR index       = 20
           **tagrew_psa_addr    = 0009EE40
           **tagrew_vir_addr    = 7009EE40
           **tagrew_phy_addr    = F009EE40
        [0-7] loq 9400 mtu 4458  oq 4000 ai 8 oi 84000002 (encaps size 4)
        cw-size 4 vlanid-rew 2
        gather A30 (bufhdr size 32 EoMPLS (Control Word) Imposition profile 81)
        2 tag: 17 18
        counters 1182, 10 reported 1182, 10.
```

```
Local OutputQ (Unicast):    Slot:5  Port:0  RED queue:0  COS queue:0
Output Q (Unicast):         Port:0          RED queue:0  COS queue:0
```

Configuring Tunnel Selection

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **pseudowire-class** *name*

Example:

```
Router(config)# pseudowire-class ts1
```

Establishes a pseudowire class with a name that you specify and enters pseudowire configuration mode.

Step 4 **encapsulation mpls**

Example:

```
Router(config-pw)# encapsulation mpls
```

Specifies the tunneling encapsulation. For AToM, the encapsulation type is mpls.

Step 5 **preferred-path** {**interface tunnel** *tunnel-number* | **peer**{*ip-address* | *host-name*}} [**disable-fallback**]

Example:

```
Router(config-pw)# preferred path peer 10.18.18.18
```

Specifies the MPLS traffic engineering tunnel or IP address or hostname to be used as the preferred path.

Step 6 **exit**

Example:

```
Router(config-pw)# exit
```

Exits from pseudowire configuration mode and enables the Tunnel Selection feature.

Step 7 **interface** *type slot / subslot / port*

Example:

```
Router(config)# interface atm1/1/0
```

Specifies an interface type and enters interface configuration mode.

Step 8 **encapsulation** *encapsulation-type*

Example:

```
Router(config-if)# encapsulation aal5
```

Specifies the encapsulation for the interface.

Step 9 **xconnect** *peer-router-id vcid* **pw-class** *name*

Example:

```
Router(config-if)# xconnect 10.0.0.1 123 pw-class ts1
```

Binds the attachment circuit to a pseudowire VC.

Examples

In the following sample output from the **show mpls l2transport vc** command includes the following information about the VCs:

- VC 101 has been assigned a preferred path called Tunnel1. The default path is disabled, because the preferred path specified that the default path should not be used if the preferred path fails.
- VC 150 has been assigned an IP address of a loopback address on PE2. The default path can be used if the preferred path fails.

Command output that is in boldface font shows the preferred path information.

```
Router# show mpls l2transport vc detail
Local interface: Gi0/0/0.1 up, line protocol up, Eth VLAN 222 up
  Destination address: 10.16.16.16, VC ID: 101, VC status: up
    Preferred path: Tunnel1,  active
    Default path: disabled
    Tunnel label: 3, next hop point2point
    Output interface: Tu1, imposed label stack {17 16}
  Create time: 00:27:31, last status change time: 00:27:31
  Signaling protocol: LDP, peer 10.16.16.16:0 up
    MPLS VC labels: local 25, remote 16
    Group ID: local 0, remote 6
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 10, send 10
    byte totals:   receive 1260, send 1300
    packet drops:  receive 0, send 0
Local interface: ATM1/0/0 up, line protocol up, ATM AAL5 0/50 up
  Destination address: 10.16.16.16, VC ID: 150, VC status: up
```

```
  Preferred path: 10.18.18.18, active
  Default path: ready
  Tunnel label: 3, next hop point2point
  Output interface: Tu2, imposed label stack {18 24}
Create time: 00:15:08, last status change time: 00:07:37
Signaling protocol: LDP, peer 10.16.16.16:0 up
  MPLS VC labels: local 26, remote 24
  Group ID: local 2, remote 0
  MTU: local 4470, remote 4470
  Remote interface description:
Sequencing: receive disabled, send disabled
VC statistics:
  packet totals: receive 0, send 0
  byte totals:   receive 0, send 0
  packet drops:  receive 0, send 0
```

Troubleshooting Tips

To debug ATM cell packing, issue the **debug atm cell-packing** command.

Configuring Tunnel Selection Using Commands Associated with L2VPN Protocol-Based Feature

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **template type pseudowire** *name*

Example:

```
Router(config)# template type pseudowire ts1
```

Creates a template pseudowire with a name that you specify and enters pseudowire configuration mode.

Step 4 **encapsulation mpls**

Example:

```
Router(config-pw)# encapsulation mpls
```

Specifies the tunneling encapsulation. For AToM, the encapsulation type is mpls.

Step 5 **preferred-path** {**interface tunnel** *tunnel-number* | **peer** {*ip-address* | *hostname*}} [**disable-fallback**]

Example:

```
Router(config-pw)# preferred path peer 10.18.18.18
```

Specifies the MPLS traffic engineering tunnel or IP address or hostname to be used as the preferred path.

Step 6 **exit**

Example:

```
Router(config-pw)# exit
```

Exits from pseudowire configuration mode and enables the Tunnel Selection feature.

Step 7 **interface** *type slot* **/** *subslot* **/** *port*[**.** *subinterface*]

Example:

```
Router(config)# interface atm1/1/0
```

Specifies an interface type and enters interface configuration mode.

Step 8 **encapsulation** *encapsulation-type*

Example:

```
Router(config-if)# encapsulation aal5
```

Specifies the encapsulation for the interface.

Step 9 **end**

Example:

```
Router(config-if)# end
```

Exits to privileged EXEC mode.

Step 10 **interface pseudowire** *number*

Example:

```
Router(config)# interface pseudowire 100
```

Specifies the pseudowire interface and enters interface configuration mode.

Step 11 **source template type pseudowire** *name*

Example:

```
Router(config-if)# source template type pseudowire ts1
```

Configures the source template of type pseudowire named ts1.

Step 12 **neighbor** *peer-address vcid-value*

Example:

```
Router(config-if)#  neighbor 10.0.0.1 123
```

Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.

Step 13 **end**

Example:

```
Router(config-if)# end
```

Exits to privileged EXEC mode.

Step 14 **l2vpn xconnect context** *context-name*

Example:

```
Router(config)# l2vpn xconnect context con1
```

Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.

Step 15 **member pseudowire** *interface-number*

Example:

```
Router(config-xconnect)# member pseudowire 100
```

Specifies a member pseudowire to form a Layer 2 VPN (L2VPN) cross connect.

Step 16 **member** *ip-address vc-id* **encapsulation mpls**

Example:

```
Router(config-xconnect)# member 10.0.0.1 123 encapsulation mpls
```

Creates the VC to transport the Layer 2 packets.

Step 17 **end**

Example:

```
Router(config-xconnect)# end
```

Exits to privileged EXEC mode.

Troubleshooting Tips using the commands associated with the L2VPN Protocol-Based CLIs feature

You can use the **debug l2vpn atom vc event** command to troubleshoot tunnel selection. For example, if the tunnel interface that is used for the preferred path is shut down, the default path is enabled. The **debug l2vpn atom vc event** command provides the following output:

```
AToM SMGR [10.2.2.2, 101]: Processing imposition update, vc_handle 62091860, update_action
 3, remote_vc_label 16
AToM SMGR [10.2.2.2, 101]: selected route no parent rewrite: tunnel not up
AToM SMGR [10.2.2.2, 101]: Imposition Programmed, Output Interface: Et3/2
```

Setting Experimental Bits with AToM

Note Only EoMPLS and CEM is supported .

Procedure

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **class-map** *class-name*

Example:

```
Router(config)# class-map class1
```

Specifies the user-defined name of the traffic class and enters class map configuration mode.

Step 4 **match any**

Example:

```
Router(config-cmap)# match any
```

Specifies that all packets will be matched. Use only the **any** keyword. Other keywords might cause unexpected results.

Step 5 **policy-map** *policy-name*

Example:

```
Router(config-cmap)# policy-map policy1
```

Specifies the name of the traffic policy to configure and enters policy-map configuration mode.

Step 6 **class** *class-name*

Example:

```
Router(config-pmap)# class class1
```

Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy and enters policy-map class configuration mode.

Step 7 **set mpls experimental** *value*

Example:

```
Router(config-pmap-c)# set mpls experimental 7
```

Designates the value to which the MPLS bits are set if the packets match the specified policy map.

Step 8 **exit**

Example:

```
Router(config-pmap-c)# exit
```

Exits policy-map class configuration mode.

Step 9 **exit**

Example:

```
Router(config-pmap)# exit
```

Exits policy-map configuration mode.

Step 10 **interface** *type slot / subslot / port*

Example:

```
Router(config)# interface atm1/0/0
```

Specifies the interface type and enters interface configuration mode.

Step 11 **service-policy input** *policy-name*

Example:

```
Router(config-if)# service-policy input policy1
```

Attaches a traffic policy to an interface.

Step 12 **end**

Example:

```
Router(config-if)# end
```

Exits to privileged EXEC mode.

Step 13 **show policy-map interface** *interface-name* [*vc* [*vpi /*] *vci*] [**dlci** *dlci*] [**input** | **output**]

Example:

```
Router# show policy-map interface serial3/0/0
```

Displays the traffic policy attached to an interface.

Enabling the Control Word

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**pseudowire-class cw_enable** **Example:** `Router(config)# pseudowire-class cw_enable`	Enters pseudowire class configuration mode.
Step 4	**encapsulation mpls** **Example:** `Router(config-pw-class)# encapsulation mpls`	Specifies the tunneling encapsulation. • For AToM, the encapsulation type is MPLS.
Step 5	**control-word** **Example:** `Router(config-pw-class)# control-word`	Enables the control word.
Step 6	**end** **Example:** `Router(config-pw-class)# end`	Exits to privileged EXEC mode.

Enabling the Control Word using the commands associated with the L2VPN Protocol-Based CLIs feature

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Router> enable`	
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface pseudowire** *number* **Example:** `Router(config)# interface pseudowire 1`	Creates an interface pseudowire with a value that you specify and enters pseudowire configuration mode.
Step 4	**encapsulation mpls** **Example:** `Router(config-pw)# encapsulation mpls`	Specifies the tunneling encapsulation. • For AToM, the encapsulation type is mpls.
Step 5	**control-word include** **Example:** `Router(config-pw)# control-word include`	Enables the control word.
Step 6	**neighbor** *peer-address vcid-value* **Example:** `Router(config-pw)# neighbor 10.0.0.1 123`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 7	**end** **Example:** `Router(config-pw)# end`	Exits to privileged EXEC mode.

Configuring MPLS AToM Remote Ethernet Port Shutdown

Note The Any Transport over MPLS (AToM): Remote Ethernet Port Shutdown feature is automatically enabled by default when an image with the feature supported is loaded on the router.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Router> enable`	
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**pseudowire-class** [*pw-class-name*] **Example:** `Router(config)# pseudowire-class eompls`	Specifies the name of a Layer 2 pseudowire class and enters pseudowire class configuration mode.
Step 4	**encapsulation mpls** **Example:** `Router(config-pw)# encapsulation mpls`	Specifies that MPLS is used as the data encapsulation method for tunneling Layer 2 traffic over the pseudowire.
Step 5	**exit** **Example:** `Router(config-pw)# exit`	Exits to global configuration mode.
Step 6	**interface** *type slot / subslot / port* [*.subinterface*] **Example:** `Router (config)# interface GigabitEthernet1/0/0`	Configures an interface type and enters interface configuration mode.
Step 7	**interface** *type slot / subslot / port* **Example:** `Router (config)# interface GigabitEthernet1/0/0`	Configures an interface type and enters interface configuration mode.
Step 8	**service instance** *number* **ethernet** *number* **Example:** `Router(config-if)# service instance 393 ethernet`	Configures an ethernet service instance on an interface and enters service instance configuration mode.
Step 9	**encapsulation** *default* **Example:** `Router(config-if-srv)# encapsulation default`	Specifies the encapsulation type for the interface, such as dot1q. **Note** Remote ethernet port shutdown is supported only with encapsulation default.

	Command or Action	Purpose
Step 10	**xconnect** *peer-ip-address vc-id pw-class pw-class-name* **Example:** `Router(config-if)# xconnect 10.1.1.1 1 pw-class eompls`	Binds an attachment circuit to a pseudowire, and configures an Any Transport over MPLS (AToM) static pseudowire.
Step 11	**no remote link failure notification** **Example:** `Router(config-if-xconn)# remote link failure notification`	Disables MPLS AToM remote link failure notification and shutdown.
Step 12	**remote link failure notification** **Example:** `Router(config-if-xconn)# remote link failure notification`	Enables MPLS AToM remote link failure notification and shutdown.
Step 13	**end** **Example:** `Router(config-if-xconn)# end`	Exits to privileged EXEC mode.

Configuring MPLS AToM Remote Ethernet Port Shutdown using the commands associated with the L2VPN Protocol-Based CLIs feature

Note The Any Transport over MPLS (AToM): Remote Ethernet Port Shutdown feature is automatically enabled by default when an image with the feature supported is loaded on the router.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**template type pseudowire** [*pseudowire-name*] **Example:** `Device(config)# template type pseudowire eompls`	Specifies the name of a Layer 2 pseudowire class and enters pseudowire class configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-pw)# encapsulation mpls`	Specifies that MPLS is used as the data encapsulation method for tunneling Layer 2 traffic over the pseudowire.
Step 5	**exit** **Example:** `Device(config-pw)# exit`	Exits to global configuration mode.
Step 6	**interface** *type slot / subslot / port* **Example:** `Device(config)# interface GigabitEthernet1/0/0`	Configures an interface type and enters interface configuration mode.
Step 7	**interface pseudowire** *number* **Example:** `Device(config-if)# interface pseudowire 100`	Specifies the pseudowire interface.
Step 8	**source template type pseudowire** **Example:** `Device(config-if)# source template type pseudowire eompls`	Configures the source template of type pseudowire named eompls.
Step 9	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)#  neighbor 10.1.1.1 1`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 10	**end** **Example:** `Device(config-if)# end`	Exits to privileged EXEC mode.

	Command or Action	Purpose
Step 11	**l2vpn xconnect context** *context-name* **Example:** `Device(config)# l2vpn xconnect context con1`	Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.
Step 12	**no remote link failure notification** **Example:** `Device(config-xconnect)# no remote link failure notification`	Disables MPLS AToM remote link failure notification and shutdown.
Step 13	**remote link failure notification** **Example:** `Device(config-xconnect)# remote link failure notification`	Enables MPLS AToM remote link failure notification and shutdown.
Step 14	**end** **Example:** `Device(config-xconnect)# end`	Exits to privileged EXEC mode.

Configuring Flow-Aware Transport (FAT) Load Balancing

Before you begin

Note that this configuration is applicable only on the NCS 4206 and NCS 4216 systems.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *slot* / *subslot* / *port* `[` . *subinterface* `]` **Example:**	Specifies the interface type and enters interface configuration mode.

	Command or Action	Purpose
	`Device(Config)# interface tengigabitethernet0/5/2`	
Step 4	**mtu** *mtu-value* **Example:** `Device(Config-if)# mtu 9216`	Specifies the MTU value for the interface. The MTU value specified at the interface level can be inherited by a subinterface.
Step 5	**no ip address** [*ip-address-mask*] [**secondary**] **Example:** `Device(Config-if)# no ip address`	Disables IP processing.
Step 6	**load-interval** *seconds* **Example:** `Device(Config-if)# load-interval 30`	Enables the length of time for which data is used to compute load statistics.
Step 7	**service instance** *id* **ethernet** **Example:** `Device(Config-if)# service instance 1 ethernet`	Configures an Ethernet service instance on an interface and enters service instance configuration mode.
Step 8	**encapsulation dot1q** *vlan-id* **Example:** `Device(Config-if-srv)# encapsulation dot1q 1`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.
Step 9	**rewrite ingress tag pop** *number* [**symmetric**] **Example:** `Device(Config-if-srv)# rewrite ingress tag pop 1 symmetric`	(Optional) Specifies the encapsulation adjustment to be performed on a frame ingressing a service instance and the tag to be removed from a packet.
Step 10	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 11	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 12	**interface pseudowire** *name* **Example:** `Device(config)# interface pseudowire 1`	Establishes a pseudowire with a name that you specify, and enters pseudowire class configuration mode.
Step 13	**encapsulation mpls** **Example:** `Device(config-pw-class)# encapsulation mpls`	Specifies the tunneling encapsulation. • For AToM, the encapsulation type is mpls.
Step 14	**neighbor** *peer-address vcid-value* **Example:** `Device(config-pw-class)# neighbor 4.4.4.4 1`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 15	**signaling protocol ldp** **Example:** `Device(config-pw-class)# signaling protocol ldp`	Specifies that the Label Distribution Protocol (LDP) is configured for the pseudowire class.
Step 16	**load-balance flow-label both** **Example:** `Device(config-pw-class)# load-balance flow-label both`	Enables the Flow-Aware Transport of MPLS Pseudowire feature and specifies how flow labels are used. We recommended that you use both as the option for flow-label. However, if you choose not to use both, you can either use **load-balance flow-label transmit** or **load-balance flow-label receive** if necessary.
Step 17	**l2vpn xconnect context** *context-name* **Example:** `Device(config-pw-class)# l2vpn xconnect context FAT1`	Creates a Layer 2 VPN (L2VPN) cross connect context and enters xconnect configuration mode.
Step 18	**member pseudowire** *interface-number* **group** *context-name* **priority** *number* **Example:** `Device(config-pw-class)# member pseudowire 1 group FAT1 priority 1`	Specifies a member pseudowire to form a Layer 2 VPN (L2VPN) cross connect.
Step 19	**member TenGigabitEthernet** *interface-number* **service-instance** *id* **Example:** `Device(config-pw-class)# member`	Specifies the location of the Gigabit Ethernetmember interface.

	Command or Action	Purpose
	`TenGigabitEthernet0/5/2 service-instance 1`	
Step 20	**end** **Example:** `Device(config-pw-class)# end`	Exits to privileged EXEC mode.
Step 21	**show l2vpn atom vc detail** **Example:** `Device# show l2vpn atom vc detail`	Displays detailed output that shows information about the flow labels configured for the pseudowire.
Step 22	**show ssm id** **Example:** `Device# show ssm id`	Displays information for all Segment Switching Manager (SSM) IDs.

Examples

The following is sample output from the **show mpls l2transport vc detail** command that shows information about the VC details:

```
Device# show mpls l2transport vc 1 detail

Local interface: Te0/5/2 up, line protocol up, Eth VLAN 1 up
  Interworking type is Ethernet
  Destination address: 10.4.4.4, VC ID: 1, VC status: up
    Output interface: BD12, imposed label stack {23 16}
    Preferred path: not configured
    Default path: active
    Next hop: 10.0.0.2
  Create time: 23:12:54, last status change time: 23:09:05
    Last label FSM state change time: 23:09:02
  Signaling protocol: LDP, peer 4.4.4.4:0 up
    Targeted Hello: 10.1.1.1(LDP Id) -> 10.4.4.4, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: not configured and not enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 27, remote 16
    Group ID: local 8, remote 8
    MTU: local 9216, remote 9216
    Remote interface description:
```

```
  Sequencing: receive disabled, send disabled
  Control Word: On
  SSO Descriptor: 10.4.4.4/1, local label: 27
  Dataplane:
    SSM segment/switch IDs: 32870/4116 (used), PWID: 1
  VC statistics:
    transit packet totals: receive 0, send 0
    transit byte totals:   receive 0, send 0
    transit packet drops:  receive 0, seq error 0, send 0
```

The following is sample output from the **show ssm id** command that shows information for all Segment Switching Manager (SSM) IDs:

```
Device# show ssm id

SSM Status: 1 switch
  Switch-ID 4116 State: Open
    Segment-ID: 168039 Type: Vlan[3]
      Switch-ID:                    4116
      Physical intf:                Local
      Allocated By:                 This CPU
      Locked By:                    SIP     [1]
      Circuit status:               UP      [1]
    Class:                         SSS
      State:                        Active
      AC Switching Context:         Te0/5/2
      SSS Info : Switch Handle 2365587479 Ckt 0x458088DC
      Interworking Eth, Encap Len 4, Boardencap Len 0, MTU 9216,
      AC Encap [4 bytes]
        8100 0001
    Class:                         ADJ
      State:                        Active
      AC Adjacency context:
      adjacency = 0x45817160 [complete] RAW TenGigabitEthernet0/5/2:1
      AC Encap [4 bytes]
        8100 0001
      1stMem: 168039 2ndMem: 0 ActMem: 168039

    Segment-ID: 32870 Type: AToM[17]
      Switch-ID:                    4116
      Allocated By:                 This CPU
      Locked By:                    SIP     [1]
    Class:                         SSS
      State:                        Active
    Class:                         ADJ
      State:                        Active
```

Limitations of FAT-PW

- Load balance does not work when flow-aware transport pseudowire is configured with remote loop-free alternate and loop-free alternate configurations with Cisco IOS XE Everest 16.5.1 release version.
- Flow-label generation algorithm is modified if the Port-channel hashing algorithm is modified using command line interface.
- Starting Cisco IOS XE Fuji 16.9.x, Flow aware transport feature (FAT) is supported on VPLS on the RSP3 module.

Configuration Examples for Any Transport over MPLS

Example: ATM over MPLS

The table below shows the configuration of ATM over MPLS on two PE routers.

Table 3: ATM over MPLS Configuration Example

PE1	PE2

mpls label protocol ldp
 mpls ldp router-id Loopback0 force
!
interface Loopback0
 ip address 10.16.12.12 255.255.255.255
!
interface ATM4/0/0
 pvc 0/100 l2transport
  encapsulation aal0
  xconnect 10.13.13.13 100 encapsulation mpls
!
interface ATM4/0/0.300 point-to-point
 no ip directed-broadcast
 no atm enable-ilmi-trap
 pvc 0/300 l2transport
  encapsulation aal0
  xconnect 10.13.13.13 300 encapsulation mpls
	mpls label protocol ldp
	 mpls ldp router-id Loopback0 force
	!
	interface Loopback0
	 ip address 10.13.13.13 255.255.255.255
	
	interface ATM4/0/0
	  pvc 0/100 l2transport
	  encapsulation aal0
	  xconnect 10.16.12.12 100 encapsulation mpls
	!
	interface ATM4/0/0.300 point-to-point
	 no ip directed-broadcast
	 no atm enable-ilmi-trap
	 pvc 0/300 l2transport
	  encapsulation aal0
	  xconnect 10.16.12.12 300 encapsulation mpls

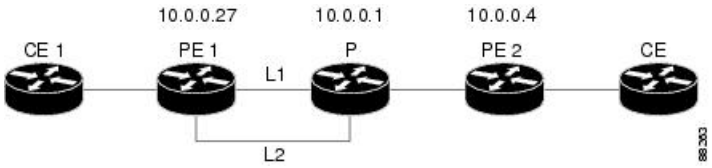
Example: Ethernet over MPLS with MPLS Traffic Engineering Fast Reroute

The following configuration example and the figure show the configuration of Ethernet over MPLS with fast reroute on AToM PE routers.

Routers PE1 and PE2 have the following characteristics:

- A TE tunnel called Tunnel41 is configured between PE1and PE2, using an explicit path through a link called L1. AToM VCs are configured to travel through the FRR-protected tunnel Tunnel41.
- The link L1 is protected by FRR, the backup tunnel is Tunnel1.
- PE2 is configured to forward the AToM traffic back to PE1 through the L2 link.

Figure 4: Fast Reroute Configuration



PE1 Configuration

```
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback1 force
!
pseudowire-class T41
 encapsulation mpls
 preferred-path interface Tunnel41 disable-fallback
!
pseudowire-class IP1
 encapsulation mpls
 preferred-path peer 10.4.0.1 disable-fallback
!
interface Loopback1
 ip address 10.0.0.27 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback1
 tunnel destination 10.0.0.1
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 10000
 tunnel mpls traffic-eng path-option 1 explicit name FRR
!
interface Tunnel41
 ip unnumbered Loopback1
 tunnel destination 10.0.0.4
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name name-1
 tunnel mpls traffic-eng fast-reroute
!
interface POS0/0/0
 description pe1name POS8/0/0
 ip address 10.1.0.2 255.255.255.252
 mpls traffic-eng tunnels
```

```
 mpls traffic-eng backup-path Tunnel1
 crc 16
 clock source internal
 pos ais-shut
 pos report lrdi
 ip rsvp bandwidth 155000 155000
!
interface POS0/3/0
 description pe1name POS10/1/0
 ip address 10.1.0.14 255.255.255.252
 mpls traffic-eng tunnels
 crc 16
 clock source internal
 ip rsvp bandwidth 155000 155000
!
interface gigabitethernet3/0/0.1
 encapsulation dot1Q 203
 xconnect 10.0.0.4 2 pw-class IP1
!
interface gigabitethernet3/0/0.2
 encapsulation dot1Q 204
 xconnect 10.0.0.4 4 pw-class T41
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 mpls traffic-eng router-id Loopback1
 mpls traffic-eng area 0
!
ip classless
ip route 10.4.0.1 255.255.255.255 Tunnel41
!
ip explicit-path name xxxx-1 enable
 next-address 10.4.1.2
 next-address 10.1.0.10
```

P Configuration

```
ip cef
mpls traffic-eng tunnels
!
interface Loopback1
 ip address 10.0.0.1 255.255.255.255
!
interface FastEthernet1/0/0
 ip address 10.4.1.2 255.255.255.0
 mpls traffic-eng tunnels
 ip rsvp bandwidth 10000 10000
!
interface POS8/0/0
 description xxxx POS0/0
 ip address 10.1.0.1 255.255.255.252
 mpls traffic-eng tunnels
 pos ais-shut
 pos report lrdi
 ip rsvp bandwidth 155000 155000
!
interface POS10/1/0
 description xxxx POS0/3
 ip address 10.1.0.13 255.255.255.252
 mpls traffic-eng tunnels
 ip rsvp bandwidth 155000 155000
!
router ospf 1
```

```
 network 10.0.0.0 0.255.255.255 area 0
 mpls traffic-eng router-id Loopback1
 mpls traffic-eng area 0
```

PE2 Configuration

```
ip cef
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback1 force
!
interface Loopback1
 ip address 10.0.0.4 255.255.255.255
!
interface loopback 2
ip address 10.4.0.1 255.255.255.255
!
interface Tunnel27
 ip unnumbered Loopback1
 tunnel destination 10.0.0.27
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name xxxx-1
!
interface FastEthernet0/0/0.2
 encapsulation dot1Q 203
 xconnect 10.0.0.27 2 encapsulation mpls
!
interface FastEthernet0/0/0.3
 encapsulation dot1Q 204
 xconnect 10.0.0.27 4 encapsulation mpls
!
interface FastEthernet1/1/0
 ip address 10.4.1.1 255.255.255.0
 mpls traffic-eng tunnels
 ip rsvp bandwidth 10000 10000
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 mpls traffic-eng router-id Loopback1
 mpls traffic-eng area 0
!
ip explicit-path name xxxx-1 enable
 next-address 10.4.1.2
 next-address 10.1.0.10
```

Example: Ethernet over MPLS with MPLS Traffic Engineering Fast Reroute Using Commands Associated with L2VPN Protocol-Based Feature

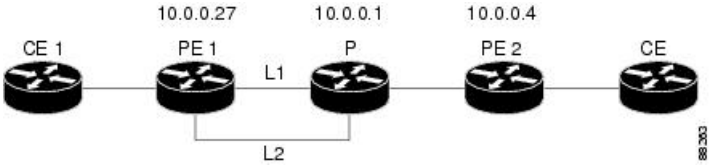
The following configuration example and the figure show the configuration of Ethernet over MPLS with fast reroute on AToM PE routers.

Routers PE1 and PE2 have the following characteristics:

- A TE tunnel called Tunnel41 is configured between PE1and PE2, using an explicit path through a link called L1. AToM VCs are configured to travel through the FRR-protected tunnel Tunnel41.
- The link L1 is protected by FRR, the backup tunnel is Tunnel1.

- PE2 is configured to forward the AToM traffic back to PE1 through the L2 link.

Figure 5: Fast Reroute Configuration



PE1 Configuration

```
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback1 force
!
template type pseudowire T41
 encapsulation mpls
 preferred-path interface Tunnel41 disable-fallback
!
template type pseudowire IP1
 encapsulation mpls
 preferred-path peer 10.4.0.1 disable-fallback
!
interface Loopback1
 ip address 10.0.0.27 255.255.255.255
!
interface Tunnel1
 ip unnumbered Loopback1
 tunnel destination 10.0.0.1
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 10000
 tunnel mpls traffic-eng path-option 1 explicit name FRR
!
interface Tunnel41
 ip unnumbered Loopback1
 tunnel destination 10.0.0.4
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name name-1
 tunnel mpls traffic-eng fast-reroute
!
interface POS0/0/0
 description pe1name POS8/0/0
 ip address 10.1.0.2 255.255.255.252
 mpls traffic-eng tunnels
 mpls traffic-eng backup-path Tunnel1
 crc 16
 clock source internal
 pos ais-shut
 pos report lrdi
 ip rsvp bandwidth 155000 155000
!
interface POS0/3/0
 description pe1name POS10/1/0
 ip address 10.1.0.14 255.255.255.252
 mpls traffic-eng tunnels
 crc 16
 clock source internal
```

```
 ip rsvp bandwidth 155000 155000
!
interface gigabitethernet3/0/0.1
 encapsulation dot1Q 203
 interface pseudowire 100
 source template type pseudowire T41
 neighbor 10.0.0.4 2
!
l2vpn xconnect context con1
!
interface gigabitethernet3/0/0.2
 encapsulation dot1Q 204
 interface pseudowire 100
 source template type pseudowire IP1
 neighbor 10.0.0.4 4
!
l2vpn xconnect context con2
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 mpls traffic-eng router-id Loopback1
 mpls traffic-eng area 0
!
ip classless
ip route 10.4.0.1 255.255.255.255 Tunnel41
!
ip explicit-path name xxxx-1 enable
 next-address 10.4.1.2
 next-address 10.1.0.10
```

P Configuration

```
ip cef
mpls traffic-eng tunnels
!
interface Loopback1
 ip address 10.0.0.1 255.255.255.255
!
interface FastEthernet1/0/0
 ip address 10.4.1.2 255.255.255.0
 mpls traffic-eng tunnels
 ip rsvp bandwidth 10000 10000
!
interface POS8/0/0
 description xxxx POS0/0
 ip address 10.1.0.1 255.255.255.252
 mpls traffic-eng tunnels
 pos ais-shut
 pos report lrdi
 ip rsvp bandwidth 155000 155000
!
interface POS10/1/0
 description xxxx POS0/3
 ip address 10.1.0.13 255.255.255.252
 mpls traffic-eng tunnels
 ip rsvp bandwidth 155000 155000
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 mpls traffic-eng router-id Loopback1
 mpls traffic-eng area 0
```

PE2 Configuration

```
ip cef
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback1 force
!
interface Loopback1
 ip address 10.0.0.4 255.255.255.255
!
interface loopback 2
ip address 10.4.0.1 255.255.255.255
!
interface Tunnel27
 ip unnumbered Loopback1
 tunnel destination 10.0.0.27
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name xxxx-1
!
interface FastEthernet0/0/0.2
 encapsulation dot1Q 203
 interface pseudowire 100
 encapsulation mpls
 neighbor 10.0.0.1 123
!
l2vpn xconnect context A
 member pseudowire 100
 member gigabitethernet 0/0/0.1
!
interface FastEthernet0/0/0.3
 encapsulation dot1Q 204
 interface pseudowire 100
 encapsulation mpls
 neighbor 10.0.0.1 123
!
l2vpn xconnect context A
 member pseudowire 100
 member gigabitethernet 0/0/0.1
!
interface FastEthernet1/1/0
 ip address 10.4.1.1 255.255.255.0
 mpls traffic-eng tunnels
 ip rsvp bandwidth 10000 10000
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
 mpls traffic-eng router-id Loopback1
 mpls traffic-eng area 0
!
ip explicit-path name xxxx-1 enable
 next-address 10.4.1.2
 next-address 10.1.0.10
```

Example: Configuring Tunnel Selection

The following example shows how to set up two preferred paths for PE1. One preferred path specifies an MPLS traffic engineering tunnel. The other preferred path specifies an IP address of a loopback address on PE2. There is a static route configured on PE1 that uses a TE tunnel to reach the IP address on PE2.

PE1 Configuration

```
mpls label protocol ldp
mpls traffic-eng tunnels
tag-switching tdp router-id Loopback0
pseudowire-class pw1
 encapsulation mpls
 preferred-path interface Tunnel1 disable-fallback
!
pseudowire-class pw2
 encapsulation mpls
 preferred-path peer 10.18.18.18
!
interface Loopback0
 ip address 10.2.2.2 255.255.255.255
 no ip directed-broadcast
 no ip mroute-cache
!
interface Tunnel1
 ip unnumbered Loopback0
 no ip directed-broadcast
 tunnel destination 10.16.16.16
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 1500
 tunnel mpls traffic-eng path-option 1 explicit name path-tu1
!
interface Tunnel2
 ip unnumbered Loopback0
 no ip directed-broadcast
 tunnel destination 10.16.16.16
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 1500
 tunnel mpls traffic-eng path-option 1 dynamic
!
interface gigabitethernet0/0/0
 no ip address
 no ip directed-broadcast
 no negotiation auto
!
interface gigabitethernet0/0/0.1
 encapsulation dot1Q 222
 no ip directed-broadcast
 xconnect 10.16.16.16 101 pw-class pw1
!
interface ATM1/0/0
 no ip address
 no ip directed-broadcast
 no atm enable-ilmi-trap
 no atm ilmi-keepalive
 pvc 0/50 l2transport
  encapsulation aal5
  xconnect 10.16.16.16 150 pw-class pw2
!
interface FastEthernet2/0/1
 ip address 10.0.0.1 255.255.255.0
 no ip directed-broadcast
 tag-switching ip
 mpls traffic-eng tunnels
 ip rsvp bandwidth 15000 15000
!
router ospf 1
```

```
 log-adjacency-changes
 network 10.0.0.0 0.0.0.255 area 0
 network 10.2.2.2 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip route 10.18.18.18 255.255.255.255 Tunnel2
!
ip explicit-path name path-tu1 enable
 next-address 10.0.0.1
 index 3 next-address 10.0.0.1
```

PE2 Configuration

```
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback0
interface Loopback0
 ip address 10.16.16.16 255.255.255.255
 no ip directed-broadcast
 no ip mroute-cache
!
interface Loopback2
 ip address 10.18.18.18 255.255.255.255
 no ip directed-broadcast
!
interface FastEthernet1/1/0
 ip address 10.0.0.2 255.255.255.0
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 no cdp enable
 ip rsvp bandwidth 15000 15000
!
interface FastEthernet1/1/1
 no ip address
 no ip directed-broadcast
 no cdp enable
!
interface FastEthernet1/1/1.1
 encapsulation dot1Q 222
 no ip directed-broadcast
 no cdp enable
 mpls l2transport route 10.2.2.2 101
!
interface ATM5/0/0
 no ip address
 no ip directed-broadcast
 no atm enable-ilmi-trap
 no atm ilmi-keepalive
 pvc 0/50 l2transport
  encapsulation aal5
  xconnect 10.2.2.2 150 encapsulation mpls
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.255 area 0
 network 10.16.16.16 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
```

Example: Configuring Tunnel Selection Using Commands Associated with L2VPN Protocol-Based Feature

The following example shows how to set up two preferred paths for PE1. One preferred path specifies an MPLS traffic engineering tunnel. The other preferred path specifies an IP address of a loopback address on PE2. There is a static route configured on PE1 that uses a TE tunnel to reach the IP address on PE2.

PE1 Configuration

```
mpls label protocol ldp
mpls traffic-eng tunnels
tag-switching tdp router-id Loopback0
template type pseudowire pw1
 encapsulation mpls
 preferred-path interface Tunnel1 disable-fallback
!
template type pseudowire pw2
 encapsulation mpls
 preferred-path peer 10.18.18.18
!
interface Loopback0
 ip address 10.2.2.2 255.255.255.255
 no ip directed-broadcast
 no ip mroute-cache
!
interface Tunnel1
 ip unnumbered Loopback0
 no ip directed-broadcast
 tunnel destination 10.16.16.16
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 1500
 tunnel mpls traffic-eng path-option 1 explicit name path-tu1
!
interface Tunnel2
 ip unnumbered Loopback0
 no ip directed-broadcast
 tunnel destination 10.16.16.16
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 1500
 tunnel mpls traffic-eng path-option 1 dynamic
!
interface gigabitethernet0/0/0
 no ip address
 no ip directed-broadcast
 no negotiation auto
!
interface gigabitethernet0/0/0.1
 encapsulation dot1Q 222
 no ip directed-broadcast
 interface pseudowire 100
 source template type pseudowire pw1
  neighbor 10.16.16.16 101
!
l2vpn xconnect context con1
!
interface ATM1/0/0
 no ip address
 no ip directed-broadcast
```

```
 no atm enable-ilmi-trap
 no atm ilmi-keepalive
 pvc 0/50 l2transport
  encapsulation aal5
 interface pseudowire 100
 source template type pseudowire pw2
 neighbor 10.16.16.16 150
!
l2vpn xconnect context con1
!
interface FastEthernet2/0/1
 ip address 10.0.0.1 255.255.255.0
 no ip directed-broadcast
 tag-switching ip
 mpls traffic-eng tunnels
 ip rsvp bandwidth 15000 15000
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.255 area 0
 network 10.2.2.2 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip route 10.18.18.18 255.255.255.255 Tunnel2
!
ip explicit-path name path-tu1 enable
 next-address 10.0.0.1
 index 3 next-address 10.0.0.1
```

PE2 Configuration

```
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback0
interface Loopback0
 ip address 10.16.16.16 255.255.255.255
 no ip directed-broadcast
 no ip mroute-cache
!
interface Loopback2
 ip address 10.18.18.18 255.255.255.255
 no ip directed-broadcast
!
interface FastEthernet1/1/0
 ip address 10.0.0.2 255.255.255.0
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 no cdp enable
 ip rsvp bandwidth 15000 15000
!
interface FastEthernet1/1/1
 no ip address
 no ip directed-broadcast
 no cdp enable
!
interface FastEthernet1/1/1.1
 encapsulation dot1Q 222
 no ip directed-broadcast
 no cdp enable
 mpls l2transport route 10.2.2.2 101
!
```

```
interface ATM5/0/0
 no ip address
 no ip directed-broadcast
 no atm enable-ilmi-trap
 no atm ilmi-keepalive
 pvc 0/50 l2transport
  encapsulation aal5
  interface pseudowire 100
   encapsulation mpls
   neighbor 10.2.2.2 150
!
l2vpn xconnect context A
  member pseudowire 100
  member GigabitEthernet0/0/0.1
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.255 area 0
 network 10.16.16.16 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
```

Example: Configuring MTU Values in xconnect Configuration Mode for L2VPN Interworking

The following example shows an L2VPN Interworking example. The PE1 router has a serial interface configured with an MTU value of 1492 bytes. The PE2 router uses xconnect configuration mode to set a matching MTU of 1492 bytes, which allows the two routers to form an interworking VC. If the PE2 router did not set the MTU value in xconnect configuration mode, the interface would be set to 1500 bytes by default and the VC would not come up.

Note L2VPN interworking is not supported on Cisco ASR 900 RSP3 Module.

PE1 Configuration

```
pseudowire-class atom-ipiw
 encapsulation mpls
 interworking ip
!
interface Loopback0
 ip address 10.1.1.151 255.255.255.255
!
interface Serial2/0/0
 mtu 1492
 no ip address
 encapsulation ppp
 no fair-queue
 serial restart-delay 0
 xconnect 10.1.1.152 123 pw-class atom-ipiw
!
interface Serial4/0/0
 ip address 10.151.100.1 255.255.255.252
 encapsulation ppp
 mpls ip
 serial restart-delay 0
```

```
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.151 0.0.0.0 area 0
 network 10.151.100.0 0.0.0.3 area 0
!
mpls ldp router-id Loopback0
```

PE2 Configuration

```
pseudowire-class atom-ipiw
 encapsulation mpls
 interworking ip
!
interface Loopback0
 ip address 10.1.1.152 255.255.255.255
!
interface FastEthernet0/0/0
 no ip address
 xconnect 10.1.1.151 123 pw-class atom-ipiw
  mtu 1492
!
interface Serial4/0/0
 ip address 10.100.152.2 255.255.255.252
 encapsulation ppp
 mpls ip
 serial restart-delay 0
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.152 0.0.0.0 area 0
 network 10.100.152.0 0.0.0.3 area 0
!
mpls ldp router-id Loopback0
```

The **show mpls l2transport binding** command shows that the MTU value for the local and remote routers is 1492 bytes.

PE1

```
Router# show mpls l2transport binding
Destination Address: 10.1.1.152,  VC ID: 123
    Local Label: 105
        Cbit: 1,    VC Type: PPP,    GroupID: 0
        MTU: 1492,   Interface Desc: n/a
        VCCV: CC Type: CW [1], RA [2]
              CV Type: LSPV [2]
    Remote Label: 205
        Cbit: 1,    VC Type: FastEthernet,    GroupID: 0
        MTU: 1492,   Interface Desc: n/a
        VCCV: CC Type: RA [2]
              CV Type: LSPV [2]
Router# show mpls l2transport vc detail
Local interface: Serial2/0/0 up, line protocol up, PPP up
  MPLS VC type is PPP, interworking type is IP
  Destination address: 10.1.1.152, VC ID: 123, VC status: up
    Output interface: Serial4/0/0, imposed label stack {1003 205}
    Preferred path: not configured
    Default path: active
    Next hop: point2point
  Create time: 00:25:29, last status change time: 00:24:54
```

```
  Signaling protocol: LDP, peer 10.1.1.152:0 up
    Targeted Hello: 10.1.1.151(LDP Id) -> 10.1.1.152
    Status TLV support (local/remote)   : enabled/supported
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: no fault
      Last local SSS circuit status rcvd: no fault
      Last local SSS circuit status sent: no fault
      Last local  LDP TLV    status sent: no fault
      Last remote LDP TLV    status rcvd: no fault
    MPLS VC labels: local 105, remote 205
    Group ID: local n/a, remote 0
    MTU: local 1492, remote 1492
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 30, send 29
    byte totals:   receive 2946, send 3364
    packet drops:  receive 0, send 0
```

PE2

```
Router# show mpls l2transport binding
Destination Address: 10.1.1.151,  VC ID: 123
    Local Label: 205
        Cbit: 1,    VC Type: FastEthernet,    GroupID: 0
        MTU: 1492,   Interface Desc: n/a
        VCCV: CC Type: RA [2]
              CV Type: LSPV [2]
    Remote Label: 105
        Cbit: 1,    VC Type: FastEthernet,    GroupID: 0
        MTU: 1492,   Interface Desc: n/a
        VCCV: CC Type: CW [1], RA [2]
              CV Type: LSPV [2]
Router# show mpls l2transport vc detail
Local interface: Fe0/0/0 up, line protocol up, FastEthernet up
  MPLS VC type is FastEthernet, interworking type is IP
  Destination address: 10.1.1.151, VC ID: 123, VC status: up
    Output interface: Se4/0/0, imposed label stack {1002 105}
    Preferred path: not configured
    Default path: active
    Next hop: point2point
  Create time: 00:25:19, last status change time: 00:25:19
  Signaling protocol: LDP, peer 10.1.1.151:0 up
    Targeted Hello: 10.1.1.152(LDP Id) -> 10.1.1.151
    Status TLV support (local/remote)   : enabled/supported
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: no fault
      Last local SSS circuit status rcvd: no fault
      Last local SSS circuit status sent: no fault
      Last local  LDP TLV    status sent: no fault
      Last remote LDP TLV    status rcvd: no fault
    MPLS VC labels: local 205, remote 105
    Group ID: local n/a, remote 0
    MTU: local 1492, remote 1492
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 29, send 30
    byte totals:   receive 2900, send 3426
    packet drops:  receive 0, send 0
```

Examples: Configuring Any Transport over MPLS (AToM) Remote Ethernet Port Shutdown

The following example shows how to enable remote Ethernet port shutdown:

```
configure terminal
!
pseudowire-class eompls
 encapsulation mpls
!
interface GigabitEthernet1/0/0
 xconnect 10.1.1.1 1 pw-class eompls
  remote link failure notification
```

The following example shows how to disable remote Ethernet port shutdown:

```
configure terminal
!
pseudowire-class eompls
 encapsulation mpls
!
interface GigabitEthernet1/0/0
 xconnect 10.1.1.1 1 pw-class eompls
  no remote link failure notification
```

The related **show** command output reports operational status for all remote L2 Tunnels by interface.

```
Router# show interface G1/0/0
GigabitEthernet1/0/0 is L2 Tunnel remote down, line protocol is up
Hardware is GigMac 4 Port GigabitEthernet, address is 0003.ff4e.12a8 (bia 0003.ff4e.12a8)
  Internet address is 10.9.9.2/16
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec, rely 255/255, load 1/255
Router# show ip interface brief
Interface              IP-Address      OK? Method Status Protocol
GigabitEthernet2/0/0 unassigned      YES NVRAM  L2 Tunnel remote down up
GigabitEthernet2/1/0 unassigned      YES NVRAM  administratively down down
```

Note Remote Ethernet port shutdown is enabled by default when EVC "default encapsulation" is configured.

Examples: Configuring Any Transport over MPLS (AToM) Remote Ethernet Port Shutdown Using Commands Associated with L2VPN Protocol-Based Feature

The following example shows how to enable remote Ethernet port shutdown:

```
configure terminal
!
template type pseudowire eompls
 encapsulation mpls
!
interface GigabitEthernet1/0/0
interface pseudowire 100
 source template type pseudowire eompls
 neighbor 10.1.1.1 1
```

```
!
l2vpn xconnect context con1
 remote link failure notification
```

The following example shows how to disable remote Ethernet port shutdown:

```
configure terminal
!
template type pseudowire eompls
 encapsulation mpls
!
interface GigabitEthernet1/0/0
 interface pseudowire 100
 source template type pseudowire eompls
 neighbor 10.1.1.1 1
!
l2vpn xconnect context con1
 no remote link failure notification
```

The related **show** command output reports operational status for all remote L2 Tunnels by interface.

```
Router# show interface G1/0/0
GigabitEthernet1/0/0 is L2 Tunnel remote down, line protocol is up
Hardware is GigMac 4 Port GigabitEthernet, address is 0003.ff4e.12a8 (bia 0003.ff4e.12a8)
  Internet address is 10.9.9.2/16
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec, rely 255/255, load 1/255
Router# show ip interface brief
Interface              IP-Address      OK? Method Status Protocol
GigabitEthernet2/0/0 unassigned      YES NVRAM  L2 Tunnel remote down up
GigabitEthernet2/1/0 unassigned      YES NVRAM  administratively down down
```

Additional References for Any Transport over MPLS

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Cisco IOS Multiprotocol Label Switching Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

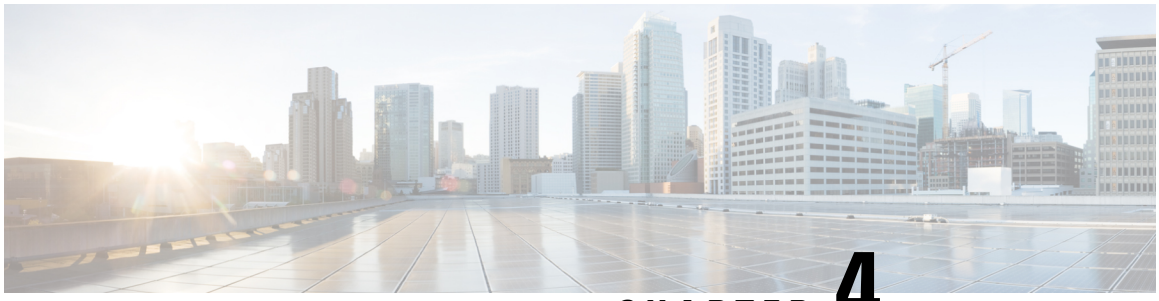
Feature Information for Any Transport over MPLS

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 4: Feature Information for Any Transport over MPLS

Feature Name	Releases	Feature Information
Any Transport over MPLS (AToM)	Cisco IOS XE Release 3.18SP	This feature was introduced on the NCS 4200 Series.



CHAPTER 4

Loop-Free Alternate Fast Reroute

Loop-Free Alternate (LFA) Fast Reroute (FRR) is a mechanism that provides local protection for unicast traffic in order to rapidly converge traffic flows around link and/or node failures.

Prerequisites for Loop-Free Alternate Fast Reroute

- Any of the following protocols must be supported for Loop-Free Alternate Fast Reroute:
  - Intermediate System-to-Intermediate System (IS-IS)
  - Open Shortest Path First (OSPF)
- While configuring ISIS protocol, **isis network point-to-point** must be configured.

Restrictions for Loop-Free Alternate Fast Reroute

- Logical interfaces namely Port-channel (PoCH) support LFA FRR and remote LFA-FRR, with a single member link. Port-channel can be used as a backup path.
- Micro loops may form due to traffic congestion.
- A Multiprotocol Label Switching (MPLS) traffic engineering (TE) tunnel cannot be used as a protected interface. However, an MPLS-TE tunnel can be a protecting (repair) interface as long as the TE tunnel is used as a primary path.

Note VPLS over TE Tunnel or TE FRR is not supported on the Cisco ASR 900 RSP3 module.

- For TDM psuedowires, the interfaces supported are CEM on OC-3.

Note This restriction is applicable only on the Cisco RSP3 Modules (NCS 4206 and NCS 4216).

- Each bridge domain interface (BDI) protected by FRR can have only one EFP.
- Remote LFA FRR provides better convergence with SFP ports rather than copper ports. As a workaround for copper ports, BFD triggered FRR can be used.
- FRR is *not* supported with POS and serial interfaces.
- Scale limit for FRR-protected global prefixes is 1500 and for layer 3 VPNs, scale limit is 4000.

Information About Loop-Free Alternate Fast Reroute

The Loop-Free Alternate (LFA) Fast Reroute (FRR) feature offers an alternative to the MPLS Traffic Engineering Fast Reroute feature to minimize packet loss due to link or node failure.

LFA FRR enables a backup route to avoid traffic loss if a network fails. The backup routes (repair paths) are precomputed and installed in the router as the backup for the primary paths. After the router detects a link or adjacent node failure, it switches to the backup path to avoid traffic loss.

LFA is a node other than the primary neighbor. Traffic is redirected to an LFA after a network failure. An LFA makes the forwarding decision without any knowledge of the failure. An LFA must neither use a failed element nor use a protecting node to forward traffic. An LFA must not cause loops. By default, LFA is enabled on all supported interfaces as long as the interface can be used as a primary path.

Advantages of using per-prefix LFAs are as follows:

- The repair path forwards traffic during transition when the primary path link is down.
- All destinations having a per-prefix LFA are protected. This leaves only a subset (a node at the far side of the failure) unprotected.

Supported Information

- LFA FRR is supported with equal cost multipath (ECMP).
- Fast Reroute triggered by Bidirectional Forwarding (BFD) is supported.
- Remote LFA tunnels are High Availability aware; hence, Stateful Switchover (SSO) compliant.

Benefits of Loop-Free Alternate Fast Reroute

- Same level of protection from traffic loss
- Simplified configuration
- Link and node protection
- Link and path protection
- LFA (loop-free alternate) paths
- Support for both IP and Label Distribution Protocol (LDP) core
- LFA FRR is supported with equal cost multipath (ECMP).
- Fast Reroute triggered by Bidirectional Forwarding (BFD).
- Remote LFA tunnels are High Availability aware; hence, Stateful Switchover (SSO) compliant.

LFA FRR and Remote LFA FRR over Bridge Domains Interfaces

The router supports bridge domain interfaces (BDI).

LFA FRR and remote LFA FRR is supported on bridge domain interfaces on the router.

IS-IS and IP FRR

When a local link fails in a network, IS-IS recomputes new primary next-hop routes for all affected prefixes. These prefixes are updated in the RIB and the Forwarding Information Base (FIB). Until the primary prefixes are updated in the forwarding plane, traffic directed towards the affected prefixes are discarded. This process can take hundreds of milliseconds.

In IP FRR, IS-IS computes LFA next-hop routes for the forwarding plane to use in case of primary path failures. LFA is computed per prefix.

When there are multiple LFAs for a given primary path, IS-IS uses a tiebreaking rule to pick a single LFA for a primary path. In case of a primary path with multiple LFA paths, prefixes are distributed equally among LFA paths.

Repair Paths

Repair paths forward traffic during a routing transition. When a link or a router fails, due to the loss of a physical layer signal, initially, only the neighboring routers are aware of the failure. All other routers in the network are unaware of the nature and location of this failure until information about this failure is propagated through a routing protocol, which may take several hundred milliseconds. It is, therefore, necessary to arrange for packets affected by the network failure to be steered to their destinations.

A router adjacent to the failed link employs a set of repair paths for packets that would have used the failed link. These repair paths are used from the time the router detects the failure until the routing transition is complete. By the time the routing transition is complete, all routers in the network revise their forwarding data and the failed link is eliminated from the routing computation.

Repair paths are precomputed in anticipation of failures so that they can be activated the moment a failure is detected.

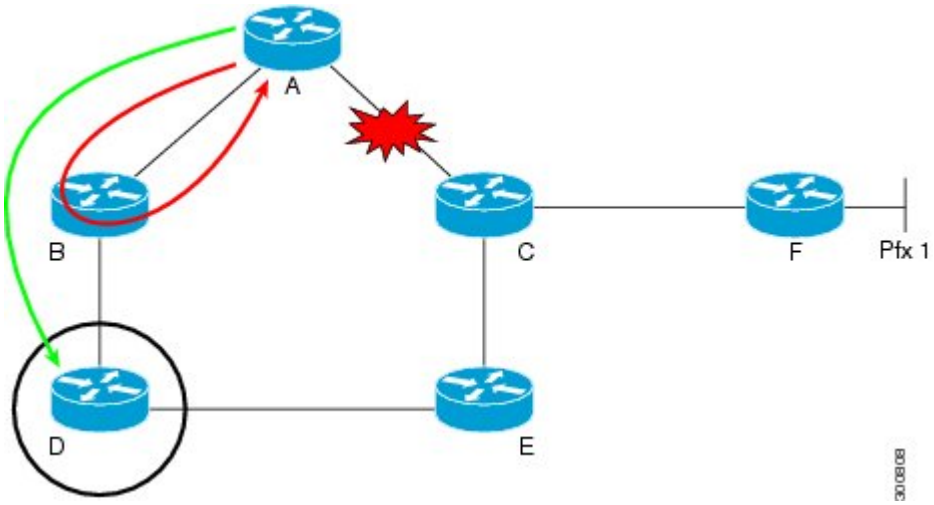
The IPv4 LFA FRR feature uses the following repair paths:

- Equal Cost Multipath (ECMP) uses a link as a member of an equal cost path-split set for a destination. The other members of the set can provide an alternative path when the link fails.
- LFA is a next-hop route that delivers a packet to its destination without looping back. Downstream paths are a subset of LFAs.

Remote LFA FRR

Some topologies (for example the commonly used ring-based topology) require protection that is not afforded by LFA FRR alone. Consider the topology shown in the figure below:

Figure 6: Remote LFA FRR with Ring Topology



The red looping arrow represents traffic that is looping immediately after a failure between node A and C (before network reconvergence). Device A tries to send traffic destined to F to next-hop B. Device B cannot be used as an LFA for prefixes advertised by nodes C and F. The actual LFA is node D. However, node D is not directly connected to the protecting node A. To protect prefixes advertised by C, node A must tunnel the packet around the failed link A-C to node D, provided that the tunnel does not traverse the failing link.

Remote LFA FRR enables you to tunnel a packet around a failed link to a remote loop-free alternate that is more than one hop away. In the figure above, the green arrow between A and D shows the tunnel that is automatically created by the remote LFA feature to bypass looping.

Remote LFA FRR for TDM and ATM Psuedowires

The Router supports two pseudowire types that utilize CEM transport: Structure-Agnostic TDM over Packet (SAToP) and Circuit Emulation Service over Packet-Switched Network (CESoPSN).

Border Gateway Protocol (BGP) Prefix-Independent Convergence (PIC) and LFA FRR Integration

Both the Labeled Border Gateway Protocol (BGP) Prefix-Independent Convergence (PIC) feature and the Loop-Free Alternate (LFA) Fast Reroute (FRR) feature can be configured together on the router.

BGP PIC is supported for bridge domain interfaces (BDI) with FRR.

Note Each bridge domain interface (BDI) protected by FRR can have only one EFP.

For information on configuring BGP PIC, see BGP PIC Edge for IP and MPLS-VPN.

Remote LFA FRR with VPLS

VPLS (Virtual Private LAN Service) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider. For information on configuring VPLS, see Configuring Virtual Private LAN Services. Starting With Cisco IOS XE Release 3.10S, Remote LFA FRR is supported with VPLS.

For information on configuring remote LFA FRR with VPLS, see How to Configure Loop-Free Alternate Fast Reroute, on page 82.

Remote LFA for MLDP

Table 5: Feature History Table

Feature Name	Release Information	Description
Remote LFA for MLDP	Cisco IOS XE Bengaluru 17.6.1	Remote Loop-Free Alternate (RLFA) based Fast Reroute (FRR) improves LFA coverage. When used with Multicast Label Distribution Protocol (MLDP) for IPv4, there's no need for an extra protocol in the control plane.

Loop-Free Alternate (LFA) FRR mechanism enables a backup route to avoid traffic loss if a network fails. The backup routes are precomputed and installed in the router as backup for primary paths. After the Cisco router detects a link or adjacent node failure, it switches to the backup path to avoid traffic loss. As the backup paths are pre-calculated and installed, switching to back up path is fast, in case of failure.

Effective Cisco IOS XE Bengaluru 17.6.1, you can enable this feature to effectively use backup routes to help with load balancing. This feature improves the LFA coverage when used with MPLS LDP (MLDP), there's no need of an extra protocol in the control plane.

Restrictions for Remote LFA for MLDP

- MLDP Node protection is not supported.
- Only Link Level protection for MLDP LFA FRR is supported.
- The detection of local/ remote link failure and switch over to the repair path is performed within 100 msec irrespective of the FRR scale.
- RLFA FRR in MLDP for IPv6 is not supported.
- RLFA FRR for Port-channel is not supported.
- Maximum supported Data Multicast Distribution Trees (MDT) for RLFA FRR is 750.
- Make Before Break (MBB) timer should be configured based on the scale of prefixes and P2MP trees configured.

- Label Distribution Protocol (LDP), Graceful Restart (GR) and MBB are mandatory to manage new and old Label Switch Path (LSP).
- Duplicate traffic is observed during cutover until GR timer expires. Duplicate traffic is dropped in the merge node and is not received by the receiver.
- RLFA over ECMP paths for mLDP is not supported.
- You must configure microloop avoidance under the IGP instance to achieve the desired convergence parameters.

Note Microloop avoidance may not work when Interior Gateway Protocol (IGP) is not receiving the interface down event to trigger the microloop avoidance timer. In such cases to avoid traffic drop due to IGP convergence, you have to configure BFD over the RLFA enabled links with 3.3-msec link down timers. This enables BFD to send link down notification within 9.9 msecs.

How to Configure Loop-Free Alternate Fast Reroute

To enable loop-free alternate fast reroute support for L2VPNs, VPLS, TDM pseudowires and VPWS, you must configure LFA FRR for the routing protocol. You can enable LFA FRR using ISIS or OSFP configurations.

- For information on configuring LFA FRR using OSPF, see OSPFv2 Loop-Free Alternate Fast Reroute in the *IP Routing: OSPF Configuration Guide*.
- For information on configuring Remote LFA FRR using OSPF, seeOSPF IPv4 Remote Loop-Free Alternate IP Fast Reroute in the *IP Routing: OSPF Configuration Guide*.
- For information on configuring Remote LFA FRR using ISIS on the Cisco ASR 903, see Configuring IS-IS Remote Loop-Free Alternate Fast Reroute, on page 82.

Configuring IS-IS Remote Loop-Free Alternate Fast Reroute

The following additional configurations are mandatory:

- **mpls ldp discovery targeted-hello accept**

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**router isis** [*area-tag*] **Example:** `Device(config)# router isis ipfrr`	Enables the IS-IS routing protocol and specifies an IS-IS process. • Enters router configuration mode.
Step 4	**fast-reroute per-prefix** {**level-1** | **level-2**} {**all** | **route-map** *route-map-name*} **Example:** `Device (config-router)# fast-reroute per-prefix level-1 all`	Enables per-prefix FRR. • Configure the **all** keyword to protect all prefixes.
Step 5	**fast-reroute remote-lfa** {**level-1** | **level-2**} **mpls-ldp** [**maximum-metric** *metric-value*] **Example:** `Device(config-router)# fast-reroute remote-lfa level-1 mpls-ldp`	Configures an FRR path that redirects traffic to a remote LFA tunnel for either level 1 or level 2 packets. • Use the **maximum-metric** *metric-value* keyword-argument pair to specify the maximum metric value required to reach the release node.
Step 6	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

Recommended Configurations ISIS

For optimal results with remote LFA FRR, it is recommended that you use the following SFP timers:

- ISIS
  - spf-interval 5 50 200
  - prc-interval 5 50 200
  - sp-gen-interval 5 50 200
  - fast-flood 10
- Globally configure the MPLS IGP hold-down timer to avoid an indefinite wait by IGP for synchronization using the **mpls ldp igp sync holdown 2000** command.

Example: Configuring IS-IS Remote Loop-Free Alternate Fast Reroute

The following example shows how to enable remote LFA FRR:

```
Router(config)# router isis
Router(config)# fast-reroute per-prefix level-1 all
Router(config)# fast-reroute per-prefix level-2 all
Router(router-config)# fast-reroute remote-lfa level-1 mpls-ldp
Router(router-config)# fast-reroute remote-lfa level-2 mpls-ldp
```

Example: Configuring Remote LFA FRR with VPLS

Example: Configuration of Remote LFA FRR with Interior Gateway Protocol (IGP)

```
router isis hp
 net 49.0101.0000.0000.0802.00
 is-type level-2-only
 ispf level-2
 metric-style wide
 fast-flood
 set-overload-bit on-startup 180
 max-lsp-lifetime 65535
 lsp-refresh-interval 65000
 spf-interval 5 50 200
 prc-interval 5 50 200
 lsp-gen-interval 5 5 200
 no hello padding
 log-adjacency-changes
 nsf cisco
 fast-reroute per-prefix level-1 all
 fast-reroute per-prefix level-2 all
 fast-reroute remote-lfa level-1 mpls-ldp
 fast-reroute remote-lfa level-2 mpls-ldp
 passive-interface Loopback0
 mpls ldp sync
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng level-2
```

Example: Configuration of Remote LFA FRR with VPLS at the interface level.

```
!
interface GigabitEthernet0/3/3
 ip address 198.51.100.1 255.255.255.0
 ip router isis hp
 logging event link-status
 load-interval 30
 negotiation auto
 mpls ip
 mpls traffic-eng tunnels
 isis network point-to-point
end
!
```

Example: Configuration of remote LFA FRR with VPLS at the global level.

```
!
l2 vfi Test-2000 manual
 vpn id 2010
 bridge-domain 2010
 neighbor 192.0.2.1 encapsulation mpls
!
```

Example: Configuration of remote LFA FRR with VPLS at Access side.

```
!
interface TenGigabitEthernet0/2/0
 no ip address
 service instance trunk 1 ethernet
  encapsulation dot1q 12-2012
  rewrite ingress tag pop 1 symmetric
  bridge-domain from-encapsulation
 !
```

How to Configure OSPF IPv4 Remote Loop-Free Alternate IP Fast Reroute

Configuring a Remote LFA Tunnel

Perform this task to configure a per-prefix LFA FRR path that redirects traffic to a remote LFA tunnel.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **Example:** `Device(config)# router ospf 10`	Enables OSPF routing and enters router configuration mode.
Step 4	**fast-reroute per-prefix remote-lfa** [**area** *area-id*] **tunnel mpls-ldp** **Example:** `Device(config-router)# fast-reroute per-prefix remote-lfa area 2 tunnel mpls-ldp`	Configures a per-prefix LFA FRR path that redirects traffic to a remote LFA tunnel via MPLS-LDP. • Use the **area** *area-id* keyword and argument to specify an area in which to enable LFA FRR.

Recommended Configurations OSPF

For optimal results with remote LFA FRR, it is recommended that you use the following SFP timers:

- timers throttle spf 50 200 5000
- timers throttle lsa 50 200 5000

- timers lsa arrival 100
- timers pacing flood 33

Note ISPF should be disabled.

Configuring the Maximum Distance to a Tunnel Endpoint

Perform this task to configure the maximum distance to the tunnel endpoint in a per-prefix LFA FRR path that redirects traffic to a remote LFA tunnel.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **Example:** `Device(config)# router ospf 10`	Enables OSPF routing and enters router configuration mode.
Step 4	**fast-reroute per-prefix remote-lfa** [**area** *area-id*] **maximum-cost** *distance* **Example:** `Device(config-router)# fast-reroute per-prefix remote-lfa area 2 maximum-cost 30`	Configures the maximum distance to the tunnel endpoint in a per-prefix LFA FRR path that redirects traffic to a remote LFA tunnel. • Use the **area** *area-id* keyword and variable to specify an area in which to enable LFA FRR.

Configuring Remote LFA FRR for MLDP

Perform this task to configure the LDP based Remote LFA FRR for MLDP.

Note Ensure to configure a carrier delay down of 1 second on all core interfaces.

You must configure the **mpls mldp forwarding recursive** command to enable recursive forwarding. Do not disable this command while configuring Remote LFA MLDP.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**mpls ldp graceful-restart** **Example:** `Router(config)# mpls ldp graceful-restart`	Enables the router to protect the LDP bindings and MPLS forwarding state during a disruption in service.
Step 4	**mpls traffic-eng tunnels** **Example:** `Router(config)# mpls traffic-eng tunnels`	Enables MPLS traffic engineering tunnel signaling on a device.
Step 5	**mpls ldp discovery targeted-hello accept** **Example:** `Router(config)#mpls ldp discovery targeted-hello accept`	Enables MPLS LDP targeted discovery on a device.
Step 6	**mpls mldp make-before-break delay 1000** **Example:** `Router(config)#mpls mldp make-before-break delay 1000`	Enables MPLS MLDP MBB on a device.
Step 7	**end** **Example:** `Router(config)# end`	Exits router configuration mode and enters privileged EXEC mode.

Configuring IGP based Remote LFA for MLDP

Perform this task to configure the IGP based Remote LFA MLDP.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router ospf***process-id* **Example:** `Router(config-router)# router ospf 10`	Enables OSPF routing and enters router configuration mode.
Step 4	**microloop avoidance rib-update-delay** *delay-time* **Example:** `Router(config-router)# microloop avoidance rib-update-delay 60000`	Specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The delay-time is in milliseconds. The range is from 1-60000.
Step 5	**fast-reroute per-prefix enable prefix-priority** *priority-level* **Example:** `Router(config-router)# fast-reroute per-prefix enable prefix-priority low`	Enables repair-path computation and selects the priority level for repair paths. Low priority specifies that all prefixes have the same eligibility for protection. High priority specifies that only high-priority prefixes are protected.
Step 6	**fast-reroute per-prefix remote-lfa**[**area** *area-id*] **tunnel mpls-ldp** **Example:** `Router(config-router)# fast-reroute per-prefix remote-lfa area 0 tunnel mpls-ldp`	Configures a per-prefix LFA FRR path that redirects traffic to a remote LFA tunnel via MPLS-LDP. Use the **area** area-id keyword and argument to specify an area in which to enable LFA FRR.
Step 7	**fast-reroute keep-all-paths** **Example:** `Router(config-router)# fast-reroute keep-all-paths`	Specifies creating a list of repair paths considered for LFA FRR.
Step 8	**end** **Example:** `Router(config)# end`	Exits router configuration mode and enters privileged EXEC mode.

Verifying Remote LFA for MLDP

Note The prefix 3.3.3.3 is mLDP Next Hop (NH) in the direction of receiver.

```
Router#show ip cef 3.3.3.3 internal
3.3.3.3/32, epoch 2, RIB[I], refcnt 9, per-destination sharing
  sources: RIB, RR, LTE
  feature space:
    IPRM: 0x00028000
    Broker: linked, distributed at 1st priority
    LFD: 3.3.3.3/32 1 local label
    dflt local label info: global/16 [0x3]
        contains path extension list
        dflt disposition chain 0x37D08728
          label implicit-null
          FRR Primary
            <primary: IP adj out of GigabitEthernet0/0/5, addr 10.1.1.2>
        dflt label switch chain 0x37D08488
          label implicit-null
          FRR Primary
            <primary: TAG adj out of GigabitEthernet0/0/5, addr 10.1.1.2>
  subblocks:
    3 RR sources [no flags]
      non-eos chain [implicit-null|17](ptr:0x37D08488)-(local:16)
  ifnums:
    GigabitEthernet0/0/5(12): 10.1.1.2
    MPLS-Remote-Lfa2(47)
  path list 3711ACA0, 3 locks, per-destination, flags 0x49 [shble, rif, hwcn]
    path 2A05F610, share 1/1, type attached nexthop, for IPv4, flags [has-rpr]
      MPLS short path extensions: [none] MOI flags = 0x21 label implicit-null
      nexthop 10.1.1.2 GigabitEthernet0/0/5 label
[implicit-null|17](ptr:0x37D08728)-(local:16), IP adj out of GigabitEthernet0/0/5, addr
10.1.1.2 379E4940
        repair: attached-nexthop 5.5.5.5 MPLS-Remote-Lfa2 (2A05F2E0)
    path 2A05F2E0, share 1/1, type attached nexthop, for IPv4, flags [rpr, rpr-only]
     nexthop 5.5.5.5 MPLS-Remote-Lfa2, repair, IP midchain out of MPLS-Remote-Lfa2 379E5740

  output chain:
    label [implicit-null|17](ptr:0x37D08728)-(local:16)
    FRR Primary (0x36EE6520)
      <primary: IP adj out of GigabitEthernet0/0/5, addr 10.1.1.2 379E4940>
      <repair:  TAG midchain out of MPLS-Remote-Lfa2 379E7340
                label 25-(local:24)
                TAG adj out of GigabitEthernet0/1/7, addr 60.1.1.2 379E4F40>
```

The following is sample output from the **show ip mfib vrf** command:

```
Router3#show ip mfib vrf vrf1 232.0.0.1
Entry Flags:    C - Directly Connected, S - Signal, IA - Inherit A flag,
                ET - Data Rate Exceeds Threshold, K - Keepalive
                DDE - Data Driven Event, HW - Hardware Installed
                ME - MoFRR ECMP entry, MNE - MoFRR Non-ECMP entry, MP - MFIB
                MoFRR Primary, RP - MRIB MoFRR Primary, P - MoFRR Primary
                MS  - MoFRR  Entry in Sync, MC - MoFRR entry in MoFRR Client,
                e   - Encap helper tunnel flag.
I/O Item Flags: IC - Internal Copy, NP - Not platform switched,
                NS - Negate Signalling, SP - Signal Present,
                A - Accept, F - Forward, RA - MRIB Accept, RF - MRIB Forward,
                MA - MFIB Accept, A2 - Accept backup,
                RA2 - MRIB Accept backup, MA2 - MFIB Accept backup
```

```
Forwarding Counts: Pkt Count/Pkts per second/Avg Pkt Size/Kbits per second
Other counts:      Total/RPF failed/Other drops
I/O Item Counts:   HW Pkt Count/FS Pkt Count/PS Pkt Count   Egress Rate in pps
VRF vrf1
 (100.1.1.2,232.0.0.1) Flags: HW
   SW Forwarding: 0/0/0/0, Other: 0/0/0
   HW Forwarding:   6106/1/100/1, Other: 0/0/0
   GigabitEthernet0/1/0 Flags: A
   Lspvif2, LSM/6, RPF-ID: *, Flags: F
     Pkts: 0/0/0    Rate: 0 pps
```

The following is sample output from the **show adjacency lspvif 2 internal** command:

```
Router#show adjacency Lspvif2 internal
Protocol Interface                Address
IP       Lspvif2                  225.0.0.0(2) (incomplete)
                                  0 packets, 0 bytes
                                  epoch 0
                                  sourced in sev-epoch 80
                                  punt (rate-limited) packets
                                  no src set
                                  L3 mtu 1500
                                  Flags (0x882)
                                  Fixup disabled
                                  HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                  IP redirect disabled
                                  Switching vector: IPv4 incomplete adj oce
                                  IPv4 MFIB wire 0x3653E020
                                  LSM-ID: 0x0
                                  Platform adj-id: 0xF8000311, 0x0, tun_qos_dpidx:0

                                  Adjacency pointer 0x379E4740
                                  Next-hop 225.0.0.0
IP       Lspvif2                  225.0.0.0(4)
                                  connectionid 6
                                  0 packets, 0 bytes
                                  epoch 0
                                  sourced in sev-epoch 80
                                  empty encap string
                                  Multicast
                                  Next chain element:
                                    replicate
                                      0: label 28
                                      label [implicit-null|17](ptr:0x37D08488)-(local:16)

                                         FRR Primary (0x31EF4540)
                                           <primary: TAG adj out of GigabitEthernet0/0/5,
 addr 10.1.1.2 379E5140>
                                           <repair:  TAG midchain out of MPLS-Remote-Lfa2
 379E7340
                                                     label 25-(local:24)
                                                     TAG adj out of GigabitEthernet0/1/7,
 addr 60.1.1.2 379E4F40>
                                    parent oce 0x37971348
                                  L3 mtu 1500
                                  Flags (0x8C6)
                                  Fixup disabled
                                  HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                  IP redirect disabled
                                  Switching vector: IPv4 midchain adj oce
                                  IPv4 MFIB wire 0x392A003C
                                  LSM-ID: 0x6
                                  MPLS subblock: flags 0x18 NSF PATH_SET
                                    Path Set Id 0x00000006   Num Paths 1   Owner MLDP
Flags 0xD   RefCount 3
```

```
                                        path recursive 3.3.3.3  label 28  MOI flags 0x1
                                        EOS output chain elements: replicate
                                          0: label 28
                                             label
[implicit-null|17](ptr:0x37D08488)-(local:16)
                                             FRR Primary (0x31EF4540)
                                             <primary: TAG adj out of GigabitEthernet0/0/5,
 addr 10.1.1.2 379E5140>
                                             <repair:  TAG midchain out of MPLS-Remote-Lfa2
 379E7340
                                                       label 25-(local:24)
                                                     TAG adj out of GigabitEthernet0/1/7,
 addr 60.1.1.2 379E4F40>
                                        NonEOS output chain elements: replicate
                                          0: label 28
                                             label
[implicit-null|17](ptr:0x37D08488)-(local:16)
                                             FRR Primary (0x31EF4540)
                                             <primary: TAG adj out of GigabitEthernet0/0/5,
 addr 10.1.1.2 379E5140>
                                             <repair:  TAG midchain out of MPLS-Remote-Lfa2
 379E7340
                                                       label 25-(local:24)
                                                     TAG adj out of GigabitEthernet0/1/7,
 addr 60.1.1.2 379E4F40>
                                        oce chain (0x37971348):
                                          replicate
                                            0: label 28
                                               label
[implicit-null|17](ptr:0x37D08488)-(local:16)
                                               FRR Primary (0x31EF4540)
                                             <primary: TAG adj out of GigabitEthernet0/0/5,
 addr 10.1.1.2 379E5140>
                                             <repair:  TAG midchain out of MPLS-Remote-Lfa2
 379E7340
                                                          label 25-(local:24)
                                                     TAG adj out of GigabitEthernet0/1/7,
 addr 60.1.1.2 379E4F40>
                                      adj-sb send-slotmask: 0x0
                                    Platform adj-id: 0x10EC, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E4540
                                    Next-hop 225.0.0.0
IP       Lspvif2                    227.0.0.0(3)
                                    connectionid 1
                                    0 packets, 0 bytes
                                    epoch 0
                                    sourced in sev-epoch 80
                                    empty encap string
                                    Inject p2mp Multicast
                                    L3 mtu 17940
                                      mtu update from interface suppressed
                                    Flags (0x8A6)
                                    Fixup disabled
                                    HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                    IP redirect disabled
                                    Switching vector: IPv4 no fixup adj oce
                                    LSM-ID: 0x1
                                    Platform adj-id: 0xF8000314, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E7540
                                    Next-hop 227.0.0.0
IPV6     Lspvif2                    FF0E::(2) (incomplete)
                                    0 packets, 0 bytes
```

```
                                    epoch 0
                                    sourced in sev-epoch 80
                                    punt (rate-limited) packets
                                    no src set
                                    L3 mtu 1500
                                    Flags (0x1882)
                                    Fixup disabled
                                    HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                    IP redirect enabled
                                    Switching vector: IPv6 adjacency oce
                                    IPv6 MFIB wire 0x3653E054
                                    LSM-ID: 0x0
                                    Platform adj-id: 0xF8000312, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E5940
                                    Next-hop FF0E::
IPV6      Lspvif2                   FF0E::(3)
                                    connectionid 5
                                    0 packets, 0 bytes
                                    epoch 0
                                    sourced in sev-epoch 80
                                    empty encap string
                                    Multicast
                                    Next chain element:
                                      label ipv6-explicit-null
                                      TAG midchain out of Lspvif2, addr FF0E::, cid: 5
379E5D40
                                      replicate
                                      parent oce 0x3791B5B0
                                    L3 mtu 1500
                                    Flags (0x18C6)
                                    Fixup disabled
                                    HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                    IP redirect enabled
                                    Switching vector: IPv6 midchain adjacency oce
                                    LSM-ID: 0x5
                                    MPLS subblock: flags 0x8 NSF
                                      Lspvif2 FF0E:: label ipv6-explicit-null
                                        oce chain (0x3791B5B0):
                                          label ipv6-explicit-null
                                          TAG midchain out of Lspvif2, addr FF0E::, cid: 5
379E5D40
                                          replicate
                                      adj-sb send-slotmask: 0x0
                                    Platform adj-id: 0x10DD, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E5F40
                                    Next-hop FF0E::
TAG       Lspvif2                   FF0E::(4)
                                    connectionid 5
                                    0 packets, 0 bytes
                                    epoch 0
                                    sourced in sev-epoch 80
                                    empty encap string
                                    Multicast
                                    Next chain element:
                                      replicate
                                      parent oce 0x37970E50
                                    L3 mtu 17940
                                    Flags (0x8C6)
                                    Fixup disabled
                                    HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                    IP redirect disabled
                                    Switching vector: MPLS midchain adjacency oce
```

```
                                    LSM-ID: 0x5
                                    MPLS subblock: flags 0x18 NSF PATH_SET
                                      Path Set Id 0x00000005   Num Paths 0   Owner MLDP
Flags 0xD   RefCount 3
                                        EOS output chain elements: replicate
                                        NonEOS output chain elements: replicate
                                        oce chain (0x37970E50):
                                          replicate
                                      adj-sb send-slotmask: 0x0
                                    Platform adj-id: 0x10DE, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E5D40
                                    Next-hop FF0E::
IPV6     Lspvif2                    FFFF::(3)
                                    connectionid 1
                                    0 packets, 0 bytes
                                    epoch 0
                                    sourced in sev-epoch 80
                                    empty encap string
                                    Inject p2mp Multicast
                                    L3 mtu 1500
                                    Flags (0x1886)
                                    Fixup disabled
                                    HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                    IP redirect enabled
                                    Switching vector: IPv6 adjacency oce
                                    LSM-ID: 0x1
                                    Platform adj-id: 0xF8000315, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E7940
                                    Next-hop FFFF::
TAG      Lspvif2                    FFFF::(3)
                                    connectionid 1
                                    0 packets, 0 bytes
                                    epoch 0
                                    sourced in sev-epoch 80
                                    empty encap string
                                    Inject p2mp Multicast
                                    L3 mtu 17940
                                    Flags (0x886)
                                    Fixup disabled
                                    HWIDB/IDB pointers 0x3831A6DC/0x3929EAA0
                                    IP redirect disabled
                                    Switching vector: MPLS adjacency oce
                                    LSM-ID: 0x1
                                    Platform adj-id: 0x10DA, 0x0, tun_qos_dpidx:0

                                    Adjacency pointer 0x379E5B40
                                    Next-hop FFFF::
Router#
```

Verifying Loop-Free Alternate Fast Reroute

Use one or more of the following commands to verify the LFA FRR configuration

- **show ip cef network-prefix internal**
- **show mpls infrastructure lfd pseudowire internal**
- **show platform hardware pp active feature cef database ipv4 network-prefix**

Example: Verifying LFA FRR with L2VPN

show ip cef internal

The following is sample output from the **show ip cef internal** command:

```
Device# show ip cef 16.16.16.16 internal
16.16.16.16/32, epoch 2, RIB[I], refcount 7, per-destination sharing
  sources: RIB, RR, LTE
  feature space:
   IPRM: 0x00028000
   Broker: linked, distributed at 1st priority
   LFD: 16.16.16.16/32 1 local label
   local label info: global/17
        contains path extension list
        disposition chain 0x3A3C1DF0
        label switch chain 0x3A3C1DF0
  subblocks:
   1 RR source [no flags]
    non-eos chain [16|44]
  ifnums:
   GigabitEthernet0/0/2(9): 7.7.7.2
   GigabitEthernet0/0/7(14): 7.7.17.9
  path 35D61070, path list 3A388FA8, share 1/1, type attached nexthop, for IPv4, flags
has-repair
    MPLS short path extensions: MOI flags = 0x20 label 16
  nexthop 7.7.7.2 GigabitEthernet0/0/2 label [16|44], adjacency IP adj out of
GigabitEthernet0/0/2, addr 7.7.7.2 35E88520
    repair: attached-nexthop 7.7.17.9 GigabitEthernet0/0/7 (35D610E0)
  path 35D610E0, path list 3A388FA8, share 1/1, type attached nexthop, for IPv4, flags
repair, repair-only
  nexthop 7.7.17.9 GigabitEthernet0/0/7, repair, adjacency IP adj out of GigabitEthernet0/0/7,
 addr 7.7.17.9 3A48A4E0
  output chain: label [16|44]
  FRR Primary (0x35D10F60)
  <primary:  TAG adj out of GigabitEthernet0/0/2, addr 7.7.7.2 35E88380>
  <repair:  TAG adj out of GigabitEthernet0/0/7, addr 7.7.17.9 3A48A340>
Rudy17#show mpls infrastructure lfd pseudowire internal
PW ID: 1VC ID: 4, Nexthop address: 16.16.16.16
SSM Class: SSS HW
Segment Count: 1
VCCV Types Supported:  cw ra ttl
Imposition details:
 Label stack {22 16}, Output interface: Gi0/0/2
 Preferred path: not configured
 Control Word: enabled, Sequencing: disabled
 FIB Non IP entry: 0x35D6CEEC
 Output chain:  AToM Imp (locks 4) label 22 label [16|44]
  FRR Primary (0x35D10F60)
  <primary:  TAG adj out of GigabitEthernet0/0/2, addr 7.7.7.2 35E88380>
Disposition details:
 Local label: 16
 Control Word: enabled, Sequencing: disabled
 SSS Switch: 3976200193
 Output chain:  mpls_eos( connid router-alert AToM Disp (locks 5)/ drop)
```

show mpls infrastructure lfd pseudowire internal

The following is sample output from the **show mpls infrastructure lfd pseudowire internal** command:

```
Device# show mpls infrastructure lfd pseudowire internal
PW ID: 1VC ID: 4, Nexthop address: 16.16.16.16
SSM Class: SSS HW
Segment Count: 1
VCCV Types Supported:  cw ra ttl
Imposition details:
 Label stack {22 16}, Output interface: Gi0/0/2
 Preferred path: not configured
 Control Word: enabled, Sequencing: disabled
 FIB Non IP entry: 0x35D6CEEC
 Output chain:  AToM Imp (locks 4) label 22 label [16|44]
  FRR Primary (0x35D10F60)
  <primary:  TAG adj out of GigabitEthernet0/0/2, addr 7.7.7.2 35E88380>
Disposition details:
 Local label: 16
 Control Word: enabled, Sequencing: disabled
 SSS Switch: 3976200193
 Output chain:  mpls_eos( connid router-alert AToM Disp (locks 5)/ drop)
```

show platform hardware pp active feature cef database

The following is sample output from the **show platform hardware pp active feature cef database** command:

```
Device# show platform hardware pp active feature cef database ipv4 16.16.16.16/32
=== CEF Prefix ===
16.16.16.16/32 -- next hop: UEA Label OCE (PI:0x104abee0, PD:0x10e6b9c8)
                Route Flags: (0)
                Handles (PI:0x104ab6e0) (PD:0x10e68140)

  HW Info:
    TCAM handle: 0x0000023f    TCAM index: 0x0000000d
    FID index  : 0x0000f804    EAID      : 0x0000808a
    MET        : 0x0000400c    FID Count : 0x00000000

=== Label OCE ===
  Label flags: 4
  Num Labels: 1
  Num Bk Labels: 1
  Out Labels: 16
  Out Backup Labels: 44
  Next OCE Type: Fast ReRoute OCE; Next OCE handle: 0x10e6f428

=== FRR OCE ===
  FRR type         : IP FRR
  FRR state        : Primary
  Primary IF's gid : 3
  Primary FID      : 0x0000f801
  FIFC entries     : 32
  PPO handle       : 0x00000000
  Next OCE         : Adjacency (0x10e63b38)
  Bkup OCE         : Adjacency (0x10e6e590)

=== Adjacency OCE ===
  Adj State: COMPLETE(0)   Address: 7.7.7.2
  Interface: GigabitEthernet0/0/2   Protocol: TAG
  mtu:1500, flags:0x0, fixups:0x0, encap_len:14
  Handles (adj_id:0x00000039) (PI:0x1041d410) (PD:0x10e63b38)
  Rewrite Str: d0:c2:82:17:8a:82:d0:c2:82:17:f2:02:88:47

  HW Info:
    FID index: 0x0000f486    EL3 index: 0x00001003    EL2 index: 0x00000000
```

```
    El2RW    : 0x00000107    MET index: 0x0000400c    EAID     : 0x00008060
    HW ADJ FLAGS: 0x40
    Hardware MAC Rewrite Str: d0:c2:82:17:8a:82:08:00:40:00:0d:02

=== Adjacency OCE ===
  Adj State: COMPLETE(0)   Address: 7.7.17.9
  Interface: GigabitEthernet0/0/7   Protocol: TAG
  mtu:1500, flags:0x0, fixups:0x0, encap_len:14
  Handles (adj_id:0x00000012) (PI:0x104acbd0) (PD:0x10e6e590)
  Rewrite Str: d0:c2:82:17:c9:83:d0:c2:82:17:f2:07:88:47

  HW Info:
    FID index: 0x0000f49d    EL3 index: 0x00001008    EL2 index: 0x00000000
    El2RW    : 0x00000111    MET index: 0x00004017    EAID     : 0x0000807d
    HW ADJ FLAGS: 0x40
    Hardware MAC Rewrite Str: d0:c2:82:17:c9:83:08:00:40:00:0d:07
```

Configuration Examples for OSPF IPv4 Remote Loop-Free Alternate IP Fast Reroute

Example: Configuring a Remote LFA Tunnel

The following example shows how to configure a remote per-prefix LFA FRR in area 2. The remote tunnel type is specified as MPLS-LDP:

```
Router(config-router)# fast-reroute per-prefix remote-lfa area 2 tunnel mpls-ldp
```

Example: Configuring the Maximum Distance to a Tunnel Endpoint

The following example shows how to set a maximum cost of 30 in area 2:

```
Router(config-router)# fast-reroute per-prefix remote-lfa area 2 maximum-cost 30
```

Example: Verifying Tunnel Interfaces Created by OSPF IPv4 Remote LFA IPFRR

The following example displays information about about tunnel interfaces created by OSPF IPv4 LFA IPFRR:

```
Router# show ip ospf fast-reroute remote-lfa tunnels

      OSPF Router with ID (192.168.1.1) (Process ID 1)
      Area with ID (0)
      Base Topology (MTID 0)

Interface MPLS-Remote-Lfa3
 Tunnel type: MPLS-LDP
 Tailend router ID: 192.168.3.3
 Termination IP address: 192.168.3.3
 Outgoing interface: Ethernet0/0
 First hop gateway: 192.168.14.4
 Tunnel metric: 20
 Protects:
  192.168.12.2 Ethernet0/1, total metric 30
```

Verifying Remote Loop-Free Alternate Fast Reroute with VPLS

Example: Verifying Remote LFA FRR with VPLS

show ip cef internal

The following is sample output from the **show ip cef internal** command:

```
Router# show ip cef 198.51.100.2/32 internal

198.51.100.2/32, epoch 2, RIB[I], refcount 7, per-destination sharing
  sources: RIB, RR, LTE
  feature space:
   IPRM: 0x00028000
   Broker: linked, distributed at 1st priority
   LFD: 198.51.100.2/32 1 local label
   local label info: global/2033
        contains path extension list
        disposition chain 0x46764E68
        label switch chain 0x46764E68
  subblocks:
   1 RR source [heavily shared]
    non-eos chain [explicit-null|70]
  ifnums:
   TenGigabitEthernet0/1/0(15): 192.0.2.10
   MPLS-Remote-Lfa2(46)
  path 44CE1290, path list 433CF8C0, share 1/1, type attached nexthop, for IPv4, flags
has-repair
    MPLS short path extensions: MOI flags = 0x21 label explicit-null
  nexthop 192.0.2.10 TenGigabitEthernet0/1/0 label [explicit-null|70], adjacency IP adj out
 of TenGigabitEthernet0/1/0, addr 192.0.2.10 404B3960
    repair: attached-nexthop 192.0.2.1 MPLS-Remote-Lfa2 (44CE1300)
  path 44CE1300, path list 433CF8C0, share 1/1, type attached nexthop, for IPv4, flags
repair, repair-only
  nexthop 192.0.2.1 MPLS-Remote-Lfa2, repair, adjacency IP midchain out of MPLS-Remote-Lfa2
 404B3B00
  output chain: label [explicit-null|70]
  FRR Primary (0x3E25CA00)
  <primary:  TAG adj out of TenGigabitEthernet0/1/0, addr 192.168.101.22 404B3CA0>
  <repair:  TAG midchain out of MPLS-Remote-Lfa2 404B37C0 label 37 TAG adj out of
GigabitEthernet0/3/3, addr 192.0.2.14 461B2F20>
```

show ip cef detail

The following is sample output from the **show ip cef detail** command:

```
Router# show ip cef 198.51.100.2/32 detail

198.51.100.2/32, epoch 2
  local label info: global/2033
  1 RR source [heavily shared]
  nexthop 192.0.2.14 TenGigabitEthernet0/1/0 label [explicit-null|70]
    repair: attached-nexthop 192.0.2.1 MPLS-Remote-Lfa2
  nexthop 192.0.2.1 MPLS-Remote-Lfa2, repair
!
```

show platform hardware pp active feature cef databas

The following is sample output from the **show platform hardware pp active feature cef database** command:

```
Router# show platform hardware pp active feature cef database ipv4 198.51.100.2/32

=== CEF Prefix ===
198.51.100.2/32 -- next hop: UEA Label OCE (PI:0x10936770, PD:0x12dd1cd8)
                Route Flags: (0)
                Handles (PI:0x109099c8) (PD:0x12945968)

  HW Info:
    TCAM handle: 0x00000266    TCAM index: 0x00000015
    FID index  : 0x00008e7f    EAID      : 0x0001d7c4
    MET        : 0x0000401c    FID Count : 0x00000000
=== Label OCE ===
  Label flags: 4
  Num Labels: 1
  Num Bk Labels: 1
  Out Labels: 0
  Out Backup Labels: 70
=== FRR OCE ===
  FRR type         : IP FRR
  FRR state        : Primary
  Primary IF's gid : 52
  Primary FID      : 0x00008cb6
  FIFC entries     : 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0
  PPO handle       : 0x00000000
  Next OCE         : Adjacency (0x130e0df0)
  Bkup OCE         : Adjacency (0x130de608)

=== Adjacency OCE ===
  Adj State: COMPLETE(0)   Address: 192.168.101.22
  Interface: TenGigabitEthernet0/1/0   Protocol: TAG
  mtu:1500, flags:0x0, fixups:0x0, encap_len:14
  Handles (adj_id:0x000016ac) (PI:0x1090cc10) (PD:0x130e0df0)
  Rewrite Str: 18:33:9d:3d:83:10:c8:f9:f9:8d:04:10:88:47
HW Info:
    FID index: 0x00008e7e    EL3 index: 0x00001034    EL2 index: 0x00000000
    El2RW    : 0x0000010d    MET index: 0x00004012    EAID     : 0x0001d7c1
    HW ADJ FLAGS: 0x40
    Hardware MAC Rewrite Str: 18:33:9d:3d:83:10:08:00:40:00:0d:10
=== Adjacency OCE ===
  Adj State: COMPLETE(0)   Address: 0
  Interface: MPLS-Remote-Lfa2   Protocol: TAG
  mtu:17940, flags:0x40, fixups:0x0, encap_len:0
  Handles (adj_id:0xf80002e8) (PI:0x10da2150) (PD:0x130de608)
  Rewrite Str:

  HW Info:
    FID index: 0x00008ca8    EL3 index: 0x0000101c    EL2 index: 0x00000000
    El2RW    : 0x00000003    MET index: 0x00004024    EAID     : 0x0001d7cb
    HW ADJ FLAGS: 0x40
    Hardware MAC Rewrite Str: 00:00:00:00:00:00:00:00:00:00:00:00

=== Label OCE ===
  Label flags: 4
  Num Labels: 1
Num Bk Labels: 1
  Out Labels: 37
  Out Backup Labels: 37
```

```
  Next OCE Type: Adjacency; Next OCE handle: 0x12943a00
=== Adjacency OCE ===
  Adj State: COMPLETE(0)   Address: 30.1.1.1
  Interface: GigabitEthernet0/3/3   Protocol: TAG
  mtu:1500, flags:0x0, fixups:0x0, encap_len:14
  Handles (adj_id:0x0000378e) (PI:0x10909738) (PD:0x12943a00)
  Rewrite Str: c8:f9:f9:8d:01:b3:c8:f9:f9:8d:04:33:88:47

  HW Info:
    FID index: 0x00008c78    EL3 index: 0x0000101c    EL2 index: 0x00000000
    El2RW    : 0x00000109    MET index: 0x0000400e    EAID     : 0x0001cf4b
    HW ADJ FLAGS: 0x40
    Hardware MAC Rewrite Str: c8:f9:f9:8d:01:b3:08:00:40:00:0d:33
```

show mpls l2transport detail

The following is sample output from the **show mpls l2transport detail** command:

```
Router# show mpls l2transport vc 2000 detail

Local interface: VFI Test-1990 vfi up
  Interworking type is Ethernet
  Destination address: 192.0.2.1, VC ID: 2000, VC status: up
    Output interface: Te0/1/0, imposed label stack {0 2217}
    Preferred path: not configured
    Default path: active
    Next hop: 192.51.100.22
  Create time: 1d08h, last status change time: 1d08h
    Last label FSM state change time: 1d08h
  Signaling protocol: LDP, peer 192.0.51.1:0 up
    Targeted Hello: 192.51.100.2(LDP Id) -> 192.51.100.200, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: not configured and not enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
      Last remote LDP TLV    status rcvd: No fault
```

Verifying Tunnel Interfaces Created by OSPF IPv4 Remote LFA IPFRR

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**show ip ospf fast-reroute remote-lfa tunnels** **Example:** `Device# show ip ospf fast-reroute remote-lfa tunnels`	Displays information about the OSPF per-prefix LFA FRR configuration.

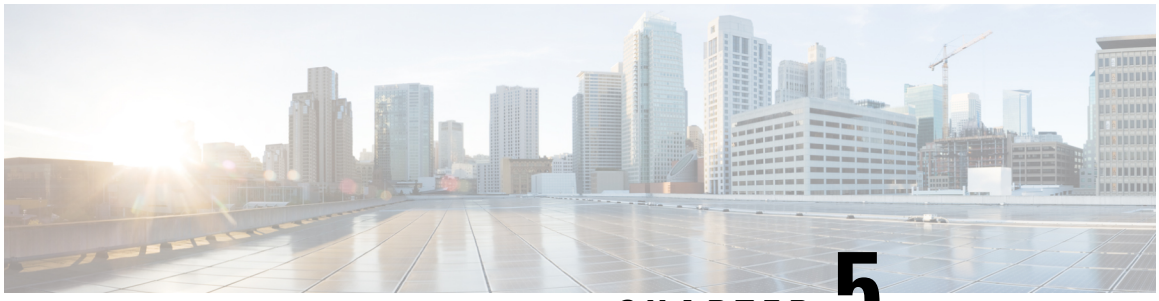
Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Multiprotocol Label Switching Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 5

Configuring Virtual Private LAN Services

Virtual Private LAN Services (VPLS) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider.

This module explains VPLS and how to configure it.

Prerequisites for Virtual Private LAN Services

Before you configure Virtual Private LAN Services (VPLS), ensure that the network is configured as follows:

- Configure IP routing in the core so that provider edge (PE) devices can reach each other via IP.
- Configure Multiprotocol Label Switching (MPLS) in the core so that a label switched path (LSP) exists between PE devices.
- Configure a loopback interface for originating and terminating Layer 2 traffic. Ensure that PE devices can access the loopback interface of the other device. Note that the loopback interface is not required in all cases. For example, tunnel selection does not need a loopback interface when VPLS is directly mapped to a traffic engineering (TE) tunnel.

Note VPLS over TE Tunnel/TE FRR is not supported.

- Identify peer PE devices and attach Layer 2 circuits to VPLS at each PE device.

Restrictions for Virtual Private LAN Services

The following general restrictions apply to all transport types under Virtual Private LAN Services (VPLS):

- Split horizon is the default configuration to avoid broadcast packet looping and to isolate Layer 2 traffic. Split horizon prevents packets received from an emulated virtual circuit (VC) from being forwarded into another emulated VC. This technique is important for creating loop-free paths in a full-meshed network.
- If you do not enable the EFP feature template, then there is no traffic flow between EFP and VFI (when EFP is with Split Horizon group and VFI is default). But when you enable the EFP feature template, then there is traffic flow between EFP and VFI because of design limitations.
- Supported maximum values:
  - Total number of virtual forwarding instances (VFIs): 4096 (4 K)
  - Total number of VFIs on the Cisco ASR 900 RSP3 module: 4096 (3584 hubs and 512 Spokes)
  - Maximum combined number of edge and the core peer provider edge (PE) devices per VFI: VPLS 250 and hierarchical VPLS (H-VPLS) 500
  - Total number of VC: 12,288 (12 K)
  - Total number of VC on the Cisco ASR 900 RSP3: 8192 (4096 EOMPLS and 4096 VFIs)
  - Maximum neighbors per VFI on the Cisco ASR 900 RSP3: 64
- Effective with Cisco IOS XE Release 3.18.2SP, the RSP3 Module only supports VPLS over Port-channel (PoCH) and bridge domain interfaces (BDI).
- VPLS over TE tunnel/TE FRR is not supported on the RSP3 Module.
- Effective Cisco IOS XE Everest 16.6.1, for VPLS to work with labeled BGP (RFC3107) on the Cisco ASR 900 RSP3 module, you must enable the following command, without which you will receive object down failure in the console:

```
router bgp [as-no]
address-family ipv4
bgp mpls-local-label
```

- Fragmentation is not supported for VPLS and VPWS traffic.

Note TTL decrements on PE imposition for VPLS traffic.

- EoMPLS/XC statistics are not supported.
- L2VPN traffic is not load balanced for inner payload src-ip, dst-ip, src-dst-ip hashing algorithms in the egress PoCh interface. We recommend you to use other hashing algorithms like src-mac, dst-mac, src-dst-mac.
- Software-based data plane is not supported.
- Auto-discovery mechanism is not supported.
- The Border Gateway Protocol (BGP) autodiscovery process does not support dynamic, hierarchical VPLS.
- Load sharing and failover on redundant customer-edge-provider-edge (CE-PE) links are not supported.
- The addition or removal of MAC addresses with Label Distribution Protocol (LDP) is not supported.

- VFI is supported only with **interface vlan** command.
- On the Cisco ASR 900 RSP3 module, VPLS imposition traffic always undergoes a recirculation in the hardware.
- Point to Multipoint (P2MP) Resource Reservation Protocol (RSVP) for MPLS Traffic Engineering (MPLS-TE) is not supported over VPLS on the Cisco RSP2 and RSP3 routers.
- Traffic drops are observed for lower sized MPLS pseudowire packets.

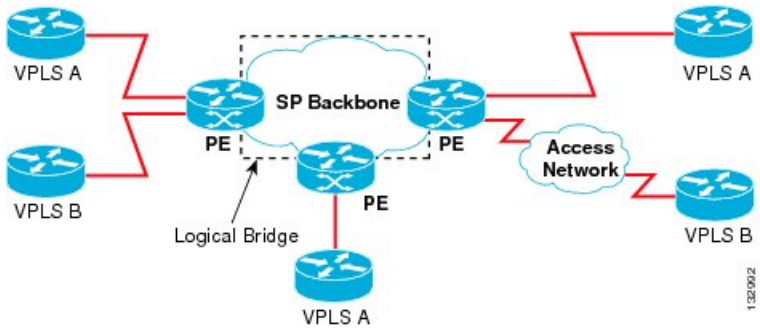
Information About Virtual Private LAN Services

VPLS Overview

Virtual Private LAN Services (VPLS) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider. From the enterprise perspective, the service provider's public network looks like one giant Ethernet LAN. For the service provider, VPLS provides an opportunity to deploy another revenue-generating service on top of the existing network without major capital expenditures. Operators can extend the operational life of equipment in their network.

VPLS uses the provider core to join multiple attachment circuits together to simulate a virtual bridge that connects the multiple attachment circuits together. From a customer point of view, there is no topology for VPLS. All customer edge (CE) devices appear to connect to a logical bridge emulated by the provider core (see the figure below).

Figure 7: VPLS Topology



Full-Mesh Configuration

A full-mesh configuration requires a full mesh of tunnel label switched paths (LSPs) between all provider edge (PE) devices that participate in Virtual Private LAN Services (VPLS). With a full mesh, signaling overhead and packet replication requirements for each provisioned virtual circuit (VC) on a PE can be high.

You set up a VPLS by first creating a virtual forwarding instance (VFI) on each participating PE device. The VFI specifies the VPN ID of a VPLS domain, the addresses of other PE devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer PE device.

The set of VFIs formed by the interconnection of the emulated VCs is called a VPLS instance; it is the VPLS instance that forms the logic bridge over a packet switched network. After the VFI has been defined, it needs to be bound to an attachment circuit to the CE device. The VPLS instance is assigned a unique VPN ID.

PE devices use the VFI to establish a full-mesh LSP of emulated VCs to all other PE devices in the VPLS instance. PE devices obtain the membership of a VPLS instance through static configuration using the Cisco IOS CLI.

A full-mesh configuration allows the PE device to maintain a single broadcast domain. When the PE device receives a broadcast, multicast, or unknown unicast packet on an attachment circuit (AC), it sends the packet out on all other ACs and emulated circuits to all other CE devices participating in that VPLS instance. The CE devices see the VPLS instance as an emulated LAN.

To avoid the problem of a packet looping in the provider core, PE devices enforce a “split-horizon” principle for emulated VCs. In a split horizon, if a packet is received on an emulated VC, it is not forwarded on any other emulated VC.

The packet forwarding decision is made by looking up the Layer 2 VFI of a particular VPLS domain.

A VPLS instance on a particular PE device receives Ethernet frames that enter on specific physical or logical ports and populates a MAC table similarly to how an Ethernet switch works. The PE device can use the MAC address to switch these frames into the appropriate LSP for delivery to the another PE device at a remote site.

If the MAC address is not available in the MAC address table, the PE device replicates the Ethernet frame and floods it to all logical ports associated with that VPLS instance, except the ingress port from which it just entered. The PE device updates the MAC table as it receives packets on specific ports and removes addresses not used for specific periods.

Static VPLS Configuration

Virtual Private LAN Services (VPLS) over Multiprotocol Label Switching-Transport Profile (MPLS-TP) tunnels allows you to deploy a multipoint-to-multipoint layer 2 operating environment over an MPLS-TP network for services such as Ethernet connectivity and multicast video. To configure static VPLS, you must specify a static range of MPLS labels using the **mpls label range** command with the **static** keyword.

H-VPLS

Hierarchical VPLS (H-VPLS) reduces signaling and replication overhead by using full-mesh and hub-and-spoke configurations. Hub-and-spoke configurations operate with split horizon to allow packets to be switched between pseudowires (PWs), effectively reducing the number of PWs between provider edge (PE) devices.

Note Split horizon is the default configuration to avoid broadcast packet looping.

Supported Features

Multipoint-to-Multipoint Support

In a multipoint-to-multipoint network, two or more devices are associated over the core network. No single device is designated as the Root node; all devices are considered as Root nodes. All frames can be exchanged directly between the nodes.

Non-Transparent Operation

A virtual Ethernet connection (VEC) can be transparent or non-transparent with respect to Ethernet protocol data units (PDUs). The VEC non-transparency allows users to have a Frame Relay-type service between Layer 3 devices.

Circuit Multiplexing

Circuit multiplexing allows a node to participate in multiple services over a single Ethernet connection. By participating in multiple services, the Ethernet connection is attached to multiple logical networks. Some examples of possible service offerings are VPN services between sites, Internet services, and third-party connectivity for intercompany communications.

MAC-Address Learning, Forwarding, and Aging

Provider edge (PE) devices must learn remote MAC addresses and directly attached MAC addresses on ports that face the external network. MAC address learning accomplishes this by deriving the topology and forwarding information from packets originating at customer sites. A timer is associated with stored MAC addresses. After the timer expires, the entry is removed from the table.

Jumbo Frame Support

Jumbo frame support provides support for frame sizes between 1548 and 9216 bytes. You use the CLI to establish the jumbo frame size for any value specified in the above range. The default value is 1500 bytes in any Layer 2/VLAN interface. You can configure jumbo frame support on a per-interface basis.

Q-in-Q Support and Q-in-Q to EoMPLS VPLS Support

With 802.1Q tunneling (Q-in-Q), the customer edge (CE) device issues VLAN-tagged packets and VPLS forwards these packets to a far-end CE device. Q-in-Q refers to the fact that one or more 802.1Q tags may be located in a packet within the interior of the network. As packets are received from a CE device, an additional VLAN tag is added to incoming Ethernet packets to segregate traffic from different CE devices. Untagged packets originating from a CE device use a single tag within the interior of the VLAN switched network, whereas previously tagged packets originating from the CE device use two or more tags.

VPLS Services

Transparent LAN Service

Transparent LAN Service (TLS) is an extension to the point-to-point port-based Ethernet over Multiprotocol Label Switching (EoMPLS), which provides bridging protocol transparency (for example, bridge protocol data units [BPDUs]) and VLAN values. Bridges see this service as an Ethernet segment. With TLS, the PE device forwards all Ethernet packets received from the customer-facing interface (including tagged and untagged packets, and BPDUs) as follows:

- To a local Ethernet interface or an emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.
- To all other local Ethernet interfaces and emulated VCs belonging to the same VPLS domain if the destination MAC address is a multicast or broadcast address or if the destination MAC address is not found in the Layer 2 forwarding table.

Note You must enable Layer 2 protocol tunneling to run the Cisco Discovery Protocol (CDP), the VLAN Trunking Protocol (VTP), and the Spanning-Tree Protocol (STP).

Ethernet Virtual Connection Service

Ethernet Virtual Connection Service (EVCS) is an extension to the point-to-point VLAN-based Ethernet over MPLS (EoMPLS) that allows devices to reach multiple intranet and extranet locations from a single physical port. With EVCS, the provider edge (PE) device forwards all Ethernet packets with a particular VLAN tag received from the customer-facing interface (excluding bridge protocol data units [BPDUs]) as follows:

- To a local Ethernet interface or to an emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.
- To all other local Ethernet interfaces and emulated VCs belonging to the same Virtual Private LAN Services (VPLS) domain if the destination MAC address is a multicast or a broadcast address or if the destination MAC address is not found in the Layer 2 forwarding table.

Note Because it has only local significance, the demultiplexing VLAN tag that identifies a VPLS domain is removed before the packet is forwarded to the outgoing Ethernet interfaces or emulated VCs.

VPLS Statistics

VPLS statistic feature supports packet and byte count in ingress and egress directions. The following are the required criteria to enable this feature:

- Metro Aggregation services license
- Special SDM template

  Use the following commands to enable or disable VPLS statistics feature:

```
sdm prefer vpls_stats_enable
sdm prefer vpls_stats_disable
```

After template configuration, the node is auto reloaded.

Restrictions

- EFP statistics is not supported when VPLS statistics is enabled.
- Transit packet drops data is not supported.
- There is a sync time of 10 seconds between the software and the hardware for fetching the statistics.
- If access rewrite is configured (pop 1), VC statistics show 4 bytes less than the actual size (in both imposition and disposition node) because pop 1 removes the VLAN header.
- VC statistics do not account LDP and VC label. It displays what is received from access in both imposition and disposition node.

Example

The following example shows a sample VPLS Statics counter output:

```
router#show mpls l2transport vc 2200 detail

Local interface: Gi0/14/2 up, line protocol up, Ethernet:100 up
  Destination address: 10.163.123.218, VC ID: 2200, VC status: up
    Output interface: Te0/7/2, imposed label stack {24022 24025}
    Preferred path: not configured
    Default path: active
    Next hop: 10.163.122.74
  Create time: 20:31:49, last status change time: 16:27:32
    Last label FSM state change time: 16:27:44
  Signaling protocol: LDP, peer 10.163.123.218:0 up
    Targeted Hello: 10.163.123.215(LDP Id) -> 10.163.123.218, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: configured and enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
     Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 110, remote 24025
    Group ID: local 40, remote 67109248
    MTU: local 9000, remote 9000
    Remote interface description: TenGigE0_0_2_3.2200
  Sequencing: receive disabled, send disabled
  Control Word: Off (configured: autosense)
  SSO Descriptor: 10.163.123.218/2200, local label: 110
  Dataplane:
    SSM segment/switch IDs: 16911/90633 (used), PWID: 71
  VC statistics:
    transit packet totals: receive 100, send 200
    transit byte totals:   receive 12800, send 25600
    transit packet drops:  receive 0, seq error 0, send 0
```

How to Configure Virtual Private LAN Services

Provisioning a Virtual Private LAN Services (VPLS) link involves provisioning the associated attachment circuit and a virtual forwarding instance (VFI) on a provider edge (PE) device.

In Cisco IOS XE Release 3.7S, the L2VPN Protocol-Based CLIs feature was introduced. This feature provides a set of processes and an improved infrastructure for developing and delivering Cisco IOS software on various Cisco platforms. This feature introduces new commands and modifies or replaces existing commands to achieve a consistent functionality across Cisco platforms and provide cross-Operating System (OS) support.

This section consists of tasks that use the commands existing prior to Cisco IOS XE Release 3.7S and a corresponding task that uses the commands introduced or modified by the L2VPN Protocol-Based CLIs feature.

Configuring PE Layer 2 Interfaces on CE Devices

You can configure the Ethernet flow point (EFP) as a Layer 2 virtual interface. You can also select tagged or untagged traffic from a customer edge (CE) device.

Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device

Note When Ethernet Virtual Connection Service (EVCS) is configured, a provider edge (PE) device forwards all Ethernet packets with a particular VLAN tag to a local Ethernet interface or emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies the service instance ID and enters service instance configuration mode.

	Command or Action	Purpose
Step 7	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this PE device.
Step 8	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance to a bridge domain instance.
Step 9	**end** **Example:** `Device(config-if-srv)# end`	Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device: Alternate Configuration

Note When Ethernet Virtual Connection Service (EVCS) is configured, the PE device forwards all Ethernet packets with a particular VLAN tag to a local Ethernet interface or an emulated virtual circuit (VC) if the destination MAC address is found in the Layer 2 forwarding table.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:**	Disables IP processing.

	Command or Action	Purpose
	`Device(config-if)# no ip address`	
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. • Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this provider edge (PE) device.
Step 8	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 11	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/0/1 service-instance 1000`	Binds a service instance to a bridge domain instance.
Step 12	**end** **Example:**	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-bdomain)# end`	

Configuring Access Ports for Untagged Traffic from a CE Device

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/0`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation untagged** **Example:** `Device(config-if-srv)# encapsulation untagged`	Defines the matching criteria to map untagged ingress Ethernet frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this provider edge (PE) device.

	Command or Action	Purpose
Step 8	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance or MAC tunnel to a bridge domain instance.
Step 9	**end** **Example:** `Device(config-if-srv)# end`	Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring Access Ports for Untagged Traffic from a CE Device: Alternate Configuration

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/4/4`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.

	Command or Action	Purpose
Step 7	**encapsulation untagged** **Example:** `Device(config-if-srv)# encapsulation untagged`	Defines the matching criteria to map untagged ingress Ethernet frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining customer edge (CE) device is on the same VLAN as this provider edge (PE) device.
Step 8	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 11	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/4/4 service-instance 1000`	Binds a service instance to a bridge domain instance.
Step 12	**end** **Example:** `Device(config-bdomain)# end`	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring Q-in-Q EFP

Note When a thread-local storage (TLS) is configured, the provider edge (PE) device forwards all Ethernet packets received from the customer edge (CE) device to all local Ethernet interfaces and emulated virtual circuits (VCs) that belong to the same Virtual Private LAN Services (VPLS) domain if the MAC address is not found in the Layer 2 forwarding table.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/2`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.
Step 7	**encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400`	Defines the matching criteria to map Q-in-Q ingress frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 8	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 100`	Binds a service instance or a MAC tunnel to a bridge domain instance.

	Command or Action	Purpose
Step 9	**end** **Example:** `Device(config-if-srv)# end`	Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring Q-in-Q EFP: Alternate Configuration

Note When a thread-local storage (TLS) is configured, the provider edge (PE) device forwards all Ethernet packets received from the customer edge (CE) device to all local Ethernet interfaces and emulated virtual circuits (VCs) belonging to the same Virtual Private LAN Services (VPLS) domain if the MAC address is not found in the Layer 2 forwarding table.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/2`	Specifies an interface and enters interface configuration mode.
Step 4	**no ip address** [*ip-address mask*] [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing.
Step 5	**negotiation auto** **Example:** `Device(config-if)# negotiation auto`	Enables the autonegotiation protocol to configure the speed, duplex, and automatic flow control of the Gigabit Ethernet interface.
Step 6	**service instance** *si-id* **ethernet** **Example:**	Specifies a service instance ID and enters service instance configuration mode.

	Command or Action	Purpose
	`Device(config-if)# service instance 10 ethernet`	
Step 7	**encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400`	Defines the matching criteria to map Q-in-Q ingress frames on an interface to the appropriate service instance. • Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 8	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 11	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/0/2 service-instance 1000`	Binds a service instance to a bridge domain instance.
Step 12	**end** **Example:** `Device(config-bdomain)# end`	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring MPLS on a PE Device

To configure Multiprotocol Label Switching (MPLS) on a provider edge (PE) device, configure the required MPLS parameters.

Note Before configuring MPLS, ensure that IP connectivity exists between all PE devices by configuring Interior Gateway Protocol (IGP), Open Shortest Path First (OSPF), or Intermediate System to Intermediate System (IS-IS) between PE devices.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mpls label protocol** {**ldp** | **tdp**} **Example:** `Device(config)# mpls label protocol ldp`	Specifies the label distribution protocol for the platform.
Step 4	**mpls ldp logging neighbor-changes** **Example:** `Device(config)# mpls ldp logging neighbor-changes`	(Optional) Generates system error logging (syslog) messages when LDP sessions go down.
Step 5	**mpls ldp discovery hello holdtime** *seconds* **Example:** `Device(config)# mpls ldp discovery hello holdtime 5`	Configures the interval between the transmission of consecutive LDP discovery hello messages or the hold time for an LDP transport connection.
Step 6	**mpls ldp router-id** *interface-type-number* [**force**] **Example:** `Device(config)# mpls ldp router-id loopback0 force`	Specifies a preferred interface for the LDP router ID.
Step 7	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring a VFI on a PE Device

The virtual forwarding interface (VFI) specifies the VPN ID of a Virtual Private LAN Services (VPLS) domain, the addresses of other provider edge (PE) devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer. Perform this task to configure a VFI:

Note Only Multiprotocol Label Switching (MPLS) encapsulation is supported.

Note You must configure BDI on the bridge domain that has the association with the VFI.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *name* **manual** **Example:** `Device(config)# l2 vfi vfi110 manual`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 110`	Configures a VPN ID for a VPLS domain. • The emulated VCs bound to this Layer 2 virtual routing and forwarding (VRF) instance use this VPN ID for signaling.
Step 5	**neighbor** *remote-router-id vc-id* {**encapsulation** *encapsulation-type* | **pw-class** *pw-name*} [**no-split-horizon**] **Example:** `Device(config-vfi)# neighbor 172.16.10.2 4 encapsulation mpls`	Specifies the type of tunnel signaling and encapsulation mechanism for each VPLS peer. **Note** Split horizon is the default configuration to avoid broadcast packet looping and to isolate Layer 2 traffic. Use the **no-split-horizon** keyword to disable split horizon and to configure multiple VCs per spoke into the same VFI.

	Command or Action	Purpose
Step 6	**bridge-domain** *bd-id* **Example:** `Device(config-vfi)# bridge-domain 100`	Specifies a bridge domain.
Step 7	**end** **Example:** `Device(config-vfi)# end`	Exits VFI configuration mode and returns to privileged EXEC mode.

Configuring a VFI on a PE Device: Alternate Configuration

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *name* **Example:** `Device(config)# l2vpn vfi context vfi110`	Establishes a L2VPN VFI between two or more separate networks, and enters VFI configuration mode.
Step 4	**vpn id** *id* **Example:** `Device(config-vfi)# vpn id 110`	Configures a VPN ID for a Virtual Private LAN Services (VPLS) domain. The emulated virtual circuits (VCs) bound to this Layer 2 virtual routing and forwarding (VRF) instance use this VPN ID for signaling.
Step 5	**member** *ip-address* [*vc-id*] **encapsulation mpls** **Example:** `Device(config-vfi)# member 172.16.10.2 4 encapsulation mpls`	Specifies the devices that form a point-to-point Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) connection and Multiprotocol Label Switching (MPLS) as the encapsulation type.
Step 6	**exit** **Example:** `Device(config-vfi)# exit`	Exits VFI configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 7	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies a bridge domain and enters bridge-domain configuration mode.
Step 8	**member vfi** *vfi-name* **Example:** `Device(config-bdomain)# member vfi vfi110`	Binds a VFI instance to a bridge domain instance.
Step 9	**end** **Example:** `Device(config-bdomain)# end`	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring Static Virtual Private LAN Services

Note Static VPLS with TP tunnel is *not* supported.

To configure static Virtual Private LAN Services (VPLS), perform the following tasks:

- Configuring a Pseudowire for Static VPLS
- Configuring VFI for Static VPLS
- Configuring a VFI for Static VPLS: Alternate Configuration
- Configuring an Attachment Circuit for Static VPLS
- Configuring an Attachment Circuit for Static VPLS: Alternate Configuration
- Configuring an MPLS-TP Tunnel for Static VPLS with TP
- Configuring a VFI for Static VPLS: Alternate Configuration

Configuring a Pseudowire for Static VPLS

Note Pseudowire for Static VPLS is *not* supported.

The configuration of pseudowires between provider edge (PE) devices helps in the successful transmission of the Layer 2 frames between PE devices.

Use the pseudowire template to configure the virtual circuit (VC) type for the virtual path identifier (VPI) pseudowire. In the following task, the pseudowire will go through a Multiprotocol Label Switching (MPLS)-Tunneling Protocol (TP) tunnel.

The pseudowire template configuration specifies the characteristics of the tunneling mechanism that is used by the pseudowires, which are:

- Encapsulation type
- Control protocol
- Payload-specific options
- Preferred path

Perform this task to configure a pseudowire template for static Virtual Private LAN Services (VPLS).

Note Ensure that you perform this task before configuring the virtual forwarding instance (VFI) peer. If the VFI peer is configured before the pseudowire class, the configuration is incomplete until the pseudowire class is configured. The **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**template type pseudowire** *name* **Example:** `Device(config)# template type pseudowire static-vpls`	Specifies the template type as pseudowire and enters template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation. • For Any Transport over MPLS (AToM), the encapsulation type is MPLS.
Step 5	**signaling protocol none** **Example:**	Specifies that no signaling protocol is configured for the pseudowire class.

	Command or Action	Purpose
	`Device(config-template)# signaling protocol none`	
Step 6	**preferred-path interface Tunnel-tp** *interface-number* **Example:** `Device(config-template)# preferred-path interface Tunnel-tp 1`	(Optional) Specifies the path that traffic uses: an MPLS Traffic Engineering (TE) tunnel or destination IP address and Domain Name Server (DNS) name.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits template configuration mode and returns to global configuration mode.
Step 8	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 1`	Establishes a pseudowire interface and enters interface configuration mode.
Step 9	**source template type pseudowire** *name* **Example:** `Device(config-if)# source template type pseudowire static-vpls`	Configures the source template type of the configured pseudowire.
Step 10	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)# neighbor 10.0.0.1 123`	Specifies the peer IP address and VC ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 11	**label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-if)# label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.
Step 12	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring VFI for Static VPLS

Note Ensure that you perform this task after configuring the pseudowire. If the VFI peer is configured before the pseudowire, the configuration is incomplete until the pseudowire is configured. The output of the **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mpls label range** *minimum-value maximum-value* [**static** *minimum-static-value maximum-static-value*] **Example:** `Device(config)# mpls label range 16 200 static 300 500`	Configures the range of local labels available for use with Multiprotocol Label Switching (MPLS) applications on packet interfaces.
Step 4	**pseudowire-class** [*pw-class-name*] **Example:** `Device(config)# pseudowire-class static_vpls`	Specifies the name of a Layer 2 pseudowire class and enters pseudowire class configuration mode.
Step 5	**encapsulation mpls** **Example:** `Device(config-pw-class)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 6	**protocol** {**l2tpv2** | **l2tpv3** | **none**} *[l2tp-class-name]* **Example:**	Specifies that no signaling protocol will be used in Layer 2 Tunneling Protocol Version 3 (L2TPv3) sessions.

	Command or Action	Purpose
	`Device(config-pw-class)# protocol none`	
Step 7	**exit** **Example:** `Device(config-pw-class)# exit`	Exits pseudowire class configuration mode and returns to global configuration mode.
Step 8	**l2 vfi** *vfi-name* **manual** **Example:** `Device(config)# l2 vfi static-vfi manual`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks, and enters Layer 2 VFI manual configuration mode.
Step 9	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 100`	Specifies the VPN ID.
Step 10	**neighbor** *ip-address* **pw-class** *pw-name* **Example:** `Device(config-vfi)# neighbor 10.3.4.4 pw-class static_vpls`	Specifies the IP address of the peer and the pseudowire class.
Step 11	**mpls label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-vfi)# mpls label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.
Step 12	**mpls control-word** **Example:** `Device(config-vfi)# mpls control-word`	(Optional) Enables the MPLS control word in an AToM static pseudowire connection.
Step 13	**neighbor** *ip-address* **pw-class** *pw-name* **Example:** `Device(config-vfi)# neighbor 2.3.4.3 pw-class static_vpls`	Specifies the IP address of the peer and the pseudowire class.
Step 14	**mpls label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-vfi)# mpls label 302 18`	Configures an AToM static pseudowire connection by defining local and remote circuit labels.

	Command or Action	Purpose
Step 15	**mpls control-word** **Example:** `Device(config-vfi)# mpls control-word`	(Optional) Enables the MPLS control word in an AToM static pseudowire connection.
Step 16	**end** **Example:** `Device(config-vfi)# end`	Exits Layer 2 VFI manual configuration mode and returns to privileged EXEC mode.

Configuring a VFI for Static VPLS: Alternate Configuration

Note Ensure that you perform this task after configuring the pseudowire. If the VFI peer is configured before the pseudowire, the configuration is incomplete until the pseudowire is configured. The output of the **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 100`	Specifies the VPN ID.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config-vfi)# exit`	Exits VFI configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface pseudowire 100`	Specifies an interface and enters interface configuration mode.
Step 7	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies an encapsulation type for tunneling Layer 2 traffic over a pseudowire.
Step 8	**neighbor** *ip-address vc-id* **Example:** `Device(config-if)# neighbor 10.3.4.4 100`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 9	**label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-if)# label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.
Step 10	**control-word** {**include** | **exclude**} **Example:** `Device(config-if)# control-word include`	(Optional) Enables the Multiprotocol Label Switching (MPLS) control word in an AToM dynamic pseudowire connection.
Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 12	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 24`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 13	**member vfi** *vfi-name* **Example:** `Device(config-bdomain)# member vfi vpls1`	Binds a service instance to a bridge domain instance.

	Command or Action	Purpose
Step 14	**end** **Example:** `Device(config-bdomain)# end`	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring an Attachment Circuit for Static VPLS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot/interface* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode. • Ensure that the interfaces between the customer edge (CE) and provider edge (PE) devices that run Ethernet over MPLS (EoMPLS) are in the same subnet. All other interfaces and backbone devices do not need to be in the same subnet.
Step 4	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 100 ethernet`	Configures an Ethernet service instance on an interface and enters service instance configuration mode.
Step 5	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. • Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 6	**rewrite ingress tag pop** *number* [**symmetric**] **Example:**	(Optional) Specifies the encapsulation adjustment to be performed on a frame

	Command or Action	Purpose
	`Device(config-if-srv)# rewrite ingress tag pop 1 symmetric`	ingressing a service instance and the tag to be removed from a packet.
Step 7	**bridge-domain** *bd-id* **Example:** `Device(config-if-srv)# bridge-domain 24`	(Optional) Binds a service instance or a MAC tunnel to a bridge domain instance.
Step 8	**end** **Example:** `Device(config-if-srv)# end`	Exits service instance configuration mode and returns to privileged EXEC mode.

Configuring an Attachment Circuit for Static VPLS: Alternate Configuration

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface gigabitethernet** *slot/interface* **Example:** `Device(config)# interface gigabitethernet 0/0/1`	Specifies an interface and enters interface configuration mode. • Ensure that the interfaces between the customer edge (CE) and provider edge (PE) devices that are running Ethernet over MPLS (EoMPLS) are in the same subnet. All other interfaces and backbone devices do not need to be in the same subnet.
Step 4	**service instance** *si-id* **ethernet** **Example:** `Device(config-if)# service instance 10 ethernet`	Specifies a service instance ID and enters service instance configuration mode.

	Command or Action	Purpose
Step 5	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if-srv)# encapsulation dot1q 200`	Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance. • Ensure that the interface on the adjoining CE device is on the same VLAN as this PE device.
Step 6	**rewrite ingress tag pop** *number* [**symmetric**] **Example:** `Device(config-if-srv)# rewrite ingress tag pop 1 symmetric`	(Optional) Specifies the encapsulation adjustment to be performed on a frame ingressing a service instance and the tag to be removed from a packet.
Step 7	**exit** **Example:** `Device(config-if-srv)# exit`	Exits service instance configuration mode and returns to interface configuration mode.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 9	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 100`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 10	**member** *interface-type-number* **service-instance** *service-id* [**split-horizon group** *group-id*] **Example:** `Device(config-bdomain)# member gigabitethernet0/0/1 service-instance 1000`	(Optional) Binds a service instance to a bridge domain instance.
Step 11	**end** **Example:** `Device(config-bdomain)# end`	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuring an MPLS-TP Tunnel for Static VPLS with TP

Note VPLS with TP/TE is not supported.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface Tunnel-tp** *number* **Example:** `Device(config)# interface Tunnel-tp 4`	Configures a Multiprotocol Label Switching (MPLS) transport profile tunnel and enters interface configuration mode. • Use the same interface as you configured for the pseudowire class.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables the IP address configuration.
Step 5	**no keepalive** **Example:** `Device(config-if)# no keepalive`	Disables the keepalive configuration.
Step 6	**tp destination** *ip-address* **Example:** `Device(config-if)# tp destination 10.22.22.22`	Configures the tunnel destination.
Step 7	**bfd** *bfd-template* **Example:** `Device(config-if)# bfd tp`	Binds a single-hop Bidirectional Forwarding Detection (BFD) template to an interface.
Step 8	**working-lsp** **Example:** `Device(config-if)# working-lsp`	Configures the working label switched path (LSP) and enters working interface configuration mode.
Step 9	**out-label** *number* **out-link** *number* **Example:**	Configures the out link and out label for the working LSP.

	Command or Action	Purpose
	`Device(config-if-working)# out-label 16 out-link 100`	
Step 10	**lsp-number** *number* **Example:** `Device(config-if-working)# lsp-number 0`	Configures the ID number for the working LSP.
Step 11	**exit** **Example:** `Device(config-if-working)# exit`	Exits working interface configuration mode and returns to interface configuration mode.
Step 12	**protect-lsp** **Example:** `Device(config-if)# protect-lsp`	Enters protection configuration mode for the label switched path (LSP) and enters protect interface configuration mode.
Step 13	**out-label** *number* **out-link** *number* **Example:** `Device(config-if-protect)# out-label 11 out-link 500`	Configures the out link and out label for the protect LSP.
Step 14	**in-label** *number* **Example:** `Device(config-if-protect)# in-label 600`	Configures the in label for the protect LSP.
Step 15	**lsp-number** *number* **Example:** `Device(config-if-protect)# lsp-number 1`	Configures the ID number for the working protect LSP.
Step 16	**exit** **Example:** `Device(config-if-protect)# exit`	Exits protect interface configuration mode and returns to interface configuration mode.
Step 17	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 18	**interface** *type number* **Example:**	Configures a interface and enters interface configuration mode.

	Command or Action	Purpose
	`Device(config-if)# interface GigabitEthernet 0/1/0`	
Step 19	**ip address** *ip-address ip-mask* **Example:** `Device(config)# ip address 10.0.0.1 255.255.255.0`	(Optional) Configures the IP address and mask if not using an IP-less core.
Step 20	**mpls tp link** *link-num* {**ipv4** *ip-address* | **tx-mac** *mac-address*} **rx-mac** *mac-address* **Example:** `Device(config-if)# mpls tp link 10 tx-mac 0100.0c99.8877 rx-mac 0100.0c99.8877`	Configures Multiprotocol Label Switching (MPLS) transport profile (TP) link parameters.
Step 21	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring a VFI for Static VPLS: Alternate Configuration

Note Ensure that you perform this task after configuring the pseudowire. If the VFI peer is configured before the pseudowire, the configuration is incomplete until the pseudowire is configured. The output of the **show running-config** command displays an error stating that configuration is incomplete.

```
Device# show running-config | sec vfi

l2 vfi config manual
 vpn id 1000
 ! Incomplete point-to-multipoint vfi config
```

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 100`	Specifies the VPN ID.
Step 5	**exit** **Example:** `Device(config-vfi)# exit`	Exits VFI configuration mode and returns to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface pseudowire 100`	Specifies an interface and enters interface configuration mode.
Step 7	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies an encapsulation type for tunneling Layer 2 traffic over a pseudowire.
Step 8	**neighbor** *ip-address vc-id* **Example:** `Device(config-if)# neighbor 10.3.4.4 100`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 9	**label** *local-pseudowire-label remote-pseudowire-label* **Example:** `Device(config-if)# label 301 17`	Configures an Any Transport over MPLS (AToM) static pseudowire connection by defining local and remote circuit labels.
Step 10	**control-word** {**include** | **exclude**} **Example:** `Device(config-if)# control-word include`	(Optional) Enables the Multiprotocol Label Switching (MPLS) control word in an AToM dynamic pseudowire connection.
Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 12	**bridge-domain** *bd-id* **Example:** `Device(config)# bridge-domain 24`	Specifies the bridge domain ID and enters bridge-domain configuration mode.
Step 13	**member vfi** *vfi-name* **Example:** `Device(config-bdomain)# member vfi vpls1`	Binds a service instance to a bridge domain instance.
Step 14	**end** **Example:** `Device(config-bdomain)# end`	Exits bridge-domain configuration mode and returns to privileged EXEC mode.

Configuration Examples for Virtual Private LAN Services

Example: Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device

This example shows how to configure the tagged traffic:

```
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# end
```

Example: Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device: Alternate Configuration

The following example shows how to configure the tagged traffic:

```
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200
Device(config-if-srv)# exit
Device(config-if)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member gigabitethernet0/0/1 service-instance 1000
Device(config-bdomain)# end
```

Example: Configuring Access Ports for Untagged Traffic from a CE Device

The following example shows how to configure access ports for untagged traffic:

```
Device(config)# interface gigabitethernet 0/0/0
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation untagged
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# end
```

The following example shows a virtual forwarding interface (VFI) configuration:

```
Device(config)# l2 vfi VPLSA manual
Device(config-vfi)# vpn id 110
Device(config-vfi)# neighbor 10.11.11.11 encapsulation mpls
Device(config-vfi)# neighbor 10.33.33.33 encapsulation mpls
Device(config-vfi)# neighbor 10.44.44.44 encapsulation mpls
Device(config-vfi)# bridge-domain 110
Device(config-vfi)# end
```

The following example shows a VFI configuration for hub and spoke.

```
Device(config)# l2 vfi VPLSB manual
Device(config-vfi)# vpn id 111
Device(config-vfi)# neighbor 10.99.99.99 encapsulation mpls
Device(config-vfi)# neighbor 10.12.12.12 encapsulation mpls
Device(config-vfi)# neighbor 10.13.13.13 encapsulation mpls no-split-horizon
Device(config-vfi)# bridge-domain 111
Device(config-vfi)# end
```

The output of the **show mpls 12transport vc** command displays various information related to a provide edge (PE) device. The VC ID in the output represents the VPN ID; the VC is identified by the combination of the destination address and the VC ID as shown in the command output. The output of the **show mpls l2transport vc detail** command displays detailed information about virtual circuits (VCs) on a PE device.

```
Device# show mpls l2transport vc 201

Local intf     Local circuit        Dest address    VC ID      Status
-------------  -------------------- --------------- ---------- ----------
VFI VPLSA      VFI                  10.11.11.11     110        UP
VFI VPLSA      VFI                  10.33.33.33     110        UP
VFI VPLSA      VFI                  10.44.44.44     110        UP
```

The following sample output from the **show vfi** command displays the VFI status:

```
Device# show vfi VPLSA

VFI name: VPLSA, state: up
  Local attachment circuits:
    Vlan2
  Neighbors connected via pseudowires:
  Peer Address     VC ID     Split-horizon
  10.11.11.11          110             Y
  10.33.33.33          110             Y
```

```
  10.44.44.44          110             Y


Device# show vfi VPLSB

VFI name: VPLSB, state: up
  Local attachment circuits:
    Vlan2
  Neighbors connected via pseudowires:
  Peer Address     VC ID     Split-horizon
  10.99.99.99       111             Y
  10.12.12.12       111             Y
  10.13.13.13       111             N
```

Example: Configuring Access Ports for Untagged Traffic from a CE Device: Alternate Configuration

The following example shows how to configure the untagged traffic.

```
Device(config)# interface GigabitEthernet 0/4/4
Device(config-if)# no ip address
Device(config-if)# negotiation auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation untagged
Device(config-if-srv)# exit
Device(config-if)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member GigabitEthernet0/4/4 service-instance 10
Device(config-if-srv)# end
```

Example: Configuring Q-in-Q EFP

The following example shows how to configure the tagged traffic.

```
Device(config)# interface GigabitEthernet 0/0/2
Device(config-if)# no ip address
Device(config-if)# negotiate auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400
Device(config-if-srv)# bridge-domain 100
Device(config-if-srv)# end
```

Use the **show spanning-tree vlan** command to verify that the ports are not in a blocked state. Use the **show vlan id** command to verify that a specific port is configured to send and receive specific VLAN traffic.

Example: Configuring Q-in-Q in EFP: Alternate Configuration

The following example shows how to configure the tagged traffic:

```
Device(config)# interface GigabitEthernet 0/4/4
Device(config-if)# no ip address
```

```
Device(config-if)# nonegotiate auto
Device(config-if)# service instance 10 ethernet
Device(config-if-srv)# encapsulation dot1q 200 second-dot1q 400
Device(config-if-srv)# exit
Device(config-if)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member GigabitEthernet0/4/4 service-instance 1000
Device(config-bdomain)# end
```

Use the **show spanning-tree vlan** command to verify that the port is not in a blocked state. Use the **show vlan id** command to verify that a specific port is configured to send and receive a specific VLAN traffic.

Example: Configuring MPLS on a PE Device

The following example shows a global Multiprotocol Label Switching (MPLS) configuration:

```
Device(config)# mpls label protocol ldp
Device(config)# mpls ldp logging neighbor-changes
Device(config)# mpls ldp discovery hello holdtime 5
Device(config)# mpls ldp router-id Loopback0 force
```

The following sample output from the **show ip cef** command displays the Label Distribution Protocol (LDP) label assigned:

```
Device# show ip cef 192.168.17.7

192.168.17.7/32, version 272, epoch 0, cached adjacency to POS4/1
0 packets, 0 bytes
  tag information set
    local tag: 8149
    fast tag rewrite with PO4/1, point2point, tags imposed: {4017}
  via 10.3.1.4, POS4/1, 283 dependencies
    next hop 10.3.1.4, POS4/1
    valid cached adjacency
    tag rewrite with PO4/1, point2point, tags imposed: {4017}
```

Example: VFI on a PE Device

The following example shows a virtual forwarding instance (VFI) configuration:

```
Device(config)# l2 vfi vfi110 manual
Device(config-vfi)# vpn id 110
Device(config-vfi)# neighbor 172.16.10.2 4 encapsulation mpls
Device(config-vfi)# neighbor 10.16.33.33 encapsulation mpls
Device(config-vfi)# neighbor 198.51.100.44 encapsulation mpls
Device(config-vfi)# bridge-domain 100
Device(config-vfi)# end
```

The following example shows a VFI configuration for a hub-and-spoke configuration:

```
Device(config)# l2 vfi VPLSA manual
Device(config-vfi)# vpn id 110
Device(config-vfi)# neighbor 10.9.9.9 encapsulation mpls
```

```
Device(config-vfi)# neighbor 192.0.2.12 encapsulation mpls
Device(config-vfi)# neighbor 203.0.113.4 encapsulation mpls no-split-horizon
Device(config-vfi)# bridge-domain 100
Device(config-vfi)# end
```

The **show mpls 12transport vc** command displays information about the provider edge (PE) device. The **show mpls l2transport vc detail** command displays detailed information about the virtual circuits (VCs) on a PE device.

```
Device# show mpls l2transport vc 201

Local intf     Local circuit        Dest address    VC ID      Status
-------------  -------------------- --------------- ---------- ----------
VFI test1      VFI                  209.165.201.1   201        UP
VFI test1      VFI                  209.165.201.2   201        UP
VFI test1      VFI                  209.165.201.3   201        UP
```

The **show vfi** *vfi-name* command displays VFI status. The VC ID in the output represents the VPN ID; the VC is identified by the combination of the destination address and the VC ID as in the example below.

```
Device# show vfi VPLS-2

VFI name: VPLS-2, state: up
  Local attachment circuits:
    Vlan2
  Neighbors connected via pseudowires:
  Peer Address     VC ID     Split-horizon
  10.1.1.1          2             Y
  10.1.1.2          2             Y
  10.2.2.3          2             N
```

Example: VFI on a PE Device: Alternate Configuration

The following example shows how to configure a virtual forwarding interface (VFI) on a provider edge (PE) device:

```
Device(config)# l2vpn vfi context vfi110
Device(config-vfi)# vpn id 110
Device(config-vfi)# member 172.16.10.2 4 encapsulation mpls
Device(config-vfi)# member 10.33.33.33 encapsulation mpls
Device(config-vfi)# member 10.44.44.44 encapsulation mpls
Device(config-vfi)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member vfi vfi110
Device(config-bdomain)# end
```

The following example shows how to configure a hub-and-spoke VFI configuration:.

```
Device(config)# l2vpn vfi context VPLSA
Device(config-vfi)# vpn id 110
Device(config-vfi)#  member 10.9.9.9 encapsulation mpls
Device(config-vfi)#  member 172.16.10.2 4 encapsulation mpls
Device(config-vfi)#  exit
Device(config)# bridge-domain 100
```

```
Device(config-bdomain)# member vfi VPLSA
Device(config-bdomain)# member GigabitEthernet0/0/0 service-instance 100
Device(config-bdomain)# member 10.33.33.33 10 encapsulation mpls
Device(config-bdomain)# end
```

The **show l2vpn atom vc** command displays information about the PE device. The command also displays information about Any Transport over MPLS (AToM) virtual circuits (VCs) and static pseudowires that are enabled to route Layer 2 packets on a device.

```
Device# show l2vpn atom vc

Local intf    Local circuit           Dest address    VC ID      Status
------------- ----------------------- --------------- ---------- ----------
Et0/0.1       Eth VLAN 101            10.0.0.2        101        UP
Et0/0.1       Eth VLAN 101            10.0.0.3        201        DOWN
```

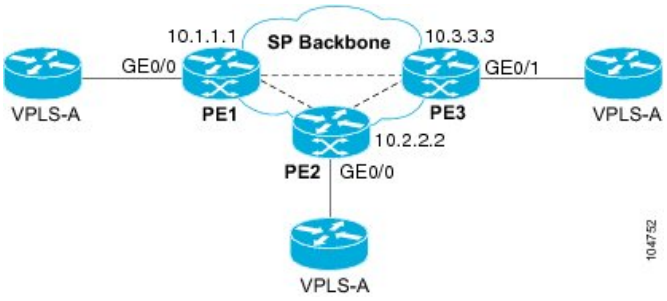
The **show l2vpn vfi** command displays the VFI status. The VC ID in the output represents the VPN ID; the VC is identified by the combination of the destination address and the VC ID as in the example below.

```
Device# show l2vpn vfi VPLS-2

Legend: RT= Route-target

VFI name: serviceCore1, State: UP, Signaling Protocol: LDP
  VPN ID: 100, VPLS-ID: 9:10, Bridge-domain vlan: 100
  RD: 9:10, RT: 10.10.10.10:150
  Pseudo-port Interface: Virtual-Ethernet1000

  Neighbors connected via pseudowires:
  Interface    Peer Address    VC ID      Discovered Router ID   Next Hop
  Pw2000       10.0.0.1        10         10.0.0.1               10.0.0.1
  Pw2001       10.0.0.2        10         10.1.1.2               10.0.0.2
  Pw2002       10.0.0.3        10         10.1.1.3               10.0.0.3
  Pw5          10.0.0.4        10         -                      10.0.0.4
```

Example: Full-Mesh VPLS Configuration

In a full-mesh configuration, each provider edge (PE) device creates a multipoint-to-multipoint forwarding relationship with all other PE devices in the Virtual Private LAN Services (VPLS) domain using a virtual forwarding interface (VFI). An Ethernet or a VLAN packet received from the customer network can be forwarded to one or more local interfaces and/or emulated virtual circuits (VCs) in the VPLS domain. To avoid a broadcast packet loop in the network, packets received from an emulated VC cannot be forwarded to any emulated VC in the VPLS domain on a PE device. Ensure that Layer 2 split horizon is enabled to avoid a broadcast packet loop in a full-mesh network.

Figure 8: Full-Mesh VPLS Configuration



PE 1 Configuration

The following examples shows how to create virtual switch instances (VSIs) and associated VCs:

```
l2 vfi PE1-VPLS-A manual
 vpn id 100
 neighbor 10.2.2.2 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
 bridge domain 100
!
interface Loopback 0
 ip address 10.1.1.1 255.255.0.0
```

The following example shows how to configure the customer edge (CE) device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface GigabitEthernet 0/0/0
 no ip address
 negotiation auto
 service instance 10 ethernet
 encapsulation dot1q 200
 bridge-domain 100
```

PE 2 Configuration

The following example shows how to create VSIs and associated VCs.

```
l2 vfi PE2-VPLS-A manual
 vpn id 100
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
 bridge domain 100
!
interface Loopback 0
 ip address 10.2.2.2 255.255.0.0
```

The following example shows how to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface GigabitEthernet 0/0/0
 no ip address
 negotiation auto
```

```
 service instance 10 ethernet
 encapsulation dot1q 200
 bridge-domain 100
```

PE 3 Configuration

The following example shows how to create VSIs and associated VCs:

```
l2 vfi PE3-VPLS-A manual
 vpn id 112
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.2.2.2 encapsulation mpls
 bridge domain 100
!
interface Loopback 0
 ip address 10.3.3.3 255.255.0.0
```

The following example shows how to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN).

```
interface GigabitEthernet 0/0/1
 no ip address
 negotiation auto
 service instance 10 ethernet
 encapsulation dot1q 200
 bridge-domain 100
!
```

The following sample output from the **show mpls l2 vc** command provides information about the status of the VC:

```
Device# show mpls l2 vc

Local intf       Local circuit        Dest address    VC ID      Status
-------------    -------------------- --------------- ---------- ----------
VFI PE1-VPLS-A   VFI                  10.2.2.2         100        UP
VFI PE1-VPLS-A   VFI                  10.3.3.3         100        UP
```

The following sample output from the **show vfi** command provides information about the VFI:

```
Device# show vfi PE1-VPLS-A

VFI name: VPLSA, state: up
  Local attachment circuits:
    Vlan200
  Neighbors connected via pseudowires:
    10.2.2.2  10.3.3.3
```

The following sample output from the **show mpls 12transport vc** command provides information about virtual circuits:

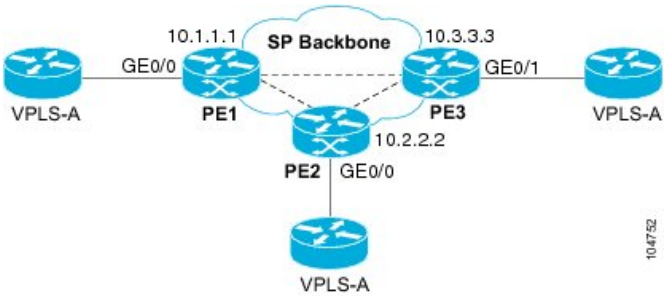
```
Device# show mpls l2transport vc detail

Local interface: VFI PE1-VPLS-A up
  Destination address: 10.2.2.2, VC ID: 100, VC status: up
    Tunnel label: imp-null, next hop point2point
```

```
    Output interface: Se2/0, imposed label stack {18}
  Create time: 3d15h, last status change time: 1d03h
  Signaling protocol: LDP, peer 10.2.2.2:0 up
    MPLS VC labels: local 18, remote 18
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 0, send 0
    byte totals:   receive 0, send 0
    packet drops:  receive 0, send 0
```

Example: Full-Mesh Configuration : Alternate Configuration

In a full-mesh configuration, each provider edge (PE) router creates a multipoint-to-multipoint forwarding relationship with all other PE routers in the Virtual Private LAN Services (VPLS) domain using a virtual forwarding interface (VFI). An Ethernet or virtual LAN (VLAN) packet received from the customer network can be forwarded to one or more local interfaces and/or emulated virtual circuits (VCs) in the VPLS domain. To avoid broadcasted packets looping in the network, no packet received from an emulated VC can be forwarded to any emulated VC of the VPLS domain on a PE router. That is, Layer 2 split horizon should always be enabled as the default in a full-mesh network.

Figure 9: VPLS Configuration Example



PE 1 Configuration

The following example shows how to create virtual switch instances (VSIs) and associated VCs and to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface gigabitethernet 0/0/0
 service instance 100 ethernet
 encap dot1q 100
 no shutdown
!
l2vpn vfi context PE1-VPLS-A
 vpn id 100
 neighbor 10.2.2.2 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
!
bridge-domain 100
 member gigabitethernet0/0/0 service-instance 100
 member vfi PE1-VPLS-A
```

PE 2 Configuration

The following example shows how to create VSIs and associated VCs and to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface gigabitethernet 0/0/0
 service instance 100 ethernet
 encap dot1q 100
 no shutdown
!
l2vpn vfi context PE2-VPLS-A
 vpn id 100
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.3.3.3 encapsulation mpls
!
bridge-domain 100
 member gigabitethernet0/0/0 service-instance 100
 member vfi PE2-VPLS-A
```

PE 3 Configuration

The following example shows how to create of the VSIs and associated VCs and to configure the CE device interface (there can be multiple Layer 2 interfaces in a VLAN):

```
interface gigabitethernet 0/0/0
 service instance 100 ethernet
 encap dot1q 100
 no shutdown
!
l2vpn vfi context PE3-VPLS-A
 vpn id 100
 neighbor 10.1.1.1 encapsulation mpls
 neighbor 10.2.2.2 encapsulation mpls
!
bridge-domain 100
 member gigabitethernet0/0/0 service-instance 100
 member vfi PE3-VPLS-A
```

The following sample output from the **show mpls l2 vc** command provides information on the status of the VC:

```
Device# show mpls l2 vc

Local intf       Local circuit  Dest address    VC ID      Status
-------------    -------------- --------------- ---------- ----------
VFI PE3-VPLS-A   VFI            10.2.2.2        100        UP
VFI PE3-VPLS-A   VFI            10.3.3.3        100        UP
```

The following sample output from the **show l2vpn vfi** command provides information about the VFI:

```
Device# show l2vpn vfi VPLS-2

Legend: RT= Route-target

VFI name: serviceCore1, State: UP, Signaling Protocol: LDP
  VPN ID: 100, VPLS-ID: 9:10, Bridge-domain vlan: 100
  RD: 9:10, RT: 10.10.10.10:150
```

```
Pseudo-port Interface: Virtual-Ethernet1000

Neighbors connected via pseudowires:
Interface    Peer Address    VC ID      Discovered Router ID   Next Hop
Pw2000       10.0.0.1        10         10.0.0.1               10.0.0.1
Pw2001       10.0.0.2        10         10.1.1.2               10.0.0.2
Pw2002       10.0.0.3        10         10.1.1.3               10.0.0.3
Pw5          10.0.0.4        10         -                      10.0.0.4
```

The following sample output from the **show l2vpn atom vc** command provides information on the virtual circuits:

```
Device# show l2vpn atom vc

Local intf    Local circuit           Dest address    VC ID      Status
------------- ----------------------- --------------- ---------- ----------
Et0/0.1       Eth VLAN 101            10.0.0.2        101        UP
Et0/0.1       Eth VLAN 101            10.0.0.3        201        DOWN
```

Flow Aware Transport (FAT) Pseudowire (PW) over VPLS

A Pseudowire (PW) load-balances traffic, between the ingress and egress PE routers, by using the Equal Cost Multiple Path (ECMP) routing technique to route packets along multiple PWs of equal cost, based on the VC label. Using multiple PWs results in wasted resources because the technique does not load balance within a PW. The distribution of multiple flows within a PW over ECMPs is a new functionality provided by FAT-PW over VPLS.

FAT-PW over VPLS uses a flow label, which is a unique identifier to distinguish a flow within the PW, and is derived from the payload of a packet. The flow label contains the End of Label Stack (EOS) bit set and inserted after the VC label and before the control word (if any). Calculation and pushing of the flow label are done by an ingress PE, which is enabled with the FAT PW configuration. The egress PE discards the flow label. For more information, see Flow-Aware Transport (FAT) Load Balancing.

Note The FAT-PW over VPLS is supported on Cisco IOS XE 16.9.1 and later. It is supported on the Cisco RSP3 module.

You can use the following commands to configure the FAT-PW over VPLS feature:

- **load-balance flow-label both**—Between two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router), PE1 and PE2, when FAT-PW is enabled, L2 traffic is load balanced in both transmit and receive directions.
- **load-balance flow-label receive**—Between two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router), PE1 and PE2, when FAT-PW is enabled, L2 traffic is load balanced only in the receive direction.
- **load-balance flow-label transmit**—Between two PE routers (that is, its head starts at the imposition PE router and its tail terminates on the disposition PE router), PE1 and PE2, when FAT-PW is enabled, L2 traffic is load balanced only in the transmit direction.

Configuring FAT-PW over VPLS

Procedure

Step 1 Configure the Bridge Domain.

Example:

```
(config)# bridge-domain 100
(config-bdomain)# member GigabitEthernet0/0/1 service-instance 123
(config-bdomain)# member vfi 100
(config-bdomain)# exit
(config)# exit
```

Step 2 Configure the service instance.

Example:

```
(config)# service instance 100 ethernet
(config)# encapsulation dot1q 100
(config)# rewrite ingress tag pop 1 symmetric
(config-if)# exit
```

Step 3 Configure the L2VPN.

Example:

```
(config)# interface pseudowire20
(config-if)# encapsulation mpls
(config-if)# neighbor 20.20.20.20 123
```

Step 4 Configure the steps to enable FAT-PW over VPLS.

Example:

```
(config-if)# load-balance flow-label ?
    both      Enable FATPW in both directions
    receive   Enable FATPW in the receive direction
    transmit  Enable FATPW in the transmit direction
(config-if)# exit
(config)# l2vpn vfi context 100
(config-cross-connect)# vpn id 100
(config-cross-connect)# member pseudowire20
(config-cross-connect)# exit
```

Restrictions for FAT-PW over VPLS

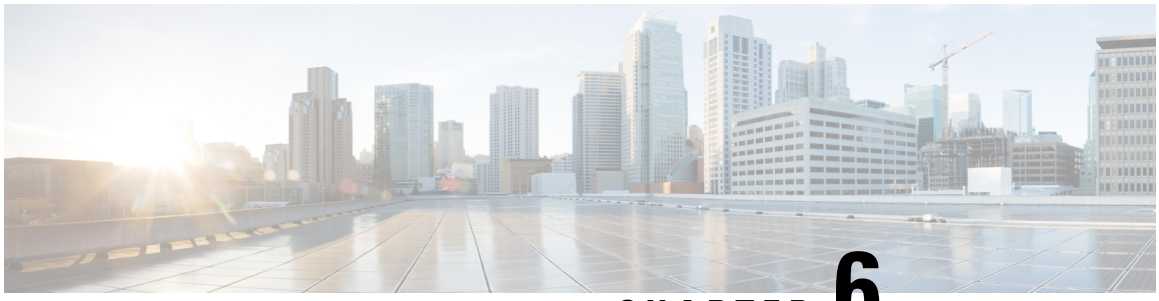
- By default, the load balance selects the **src-dst-mac-ip four-tuple** (Source MAC Address, Destination MAC Address, Source IP, and Destination IP) hash method to generate a unique flow label in the VPLS implementation.
- The RSP3 module cannot control the selection of the tuple hash method using the **load-balance flow ethernet** *both* command.
- FAT-PW cannot be enabled if VPLS is in autodiscovery mode because the load balance CLI is available only on the pseudowire interface, and it cannot be configured with autodiscovery.

- FAT-PW is not supported with the **l2 vfi** *name* manual model.
- If one of the nodes runs the 16.7.1 image that supports FAT-PW over EoMPLS, the FAT-PW negotiation is enabled for VPLS, however, there will be a considerable traffic drop. Therefore, it is recommended to run the 16.9.1 image or later for FAT-PW over VPLS.
- FAT-PW over VPLS is not supported with BGP signaling.
- Due to the existing design limitation, load balancing based on a flow-label does not work when RSP3 is deployed at the P node where rLFA/LFA configurations are present.
- Routed FAT-PW is not supported.
- Load balancing is not supported with DHCP packets.
- There is no change in the existing 4k scale number with respect to VPLS.
- The FAT-PW feature configuration is available only under the new configuration model. Therefore, all restrictions that are applicable for the new configuration model are also applicable for this feature.

Verifying FAT-PW over VPLS

Use the **show platform hardware pp active pw vpls** command to verify if the FAT flow label has been signaled and the direction of the load balancing—imposition or disposition.

```
pw    : VFI1            bdomain         : 50          vsi       : 0x15
peer_ip         : 4.4.4.4         vc_id           : 1           has_cw    : 0
STP             : FWD             status          : Disabled    sh_group  : 0
local_label     : 18              remote_label    : 19          sh_type   : Hub
imp_oce         : 0x23DBECE4      disp_oce        : 0x23DBEDC4  label_oce : 0x23DBF19C
pwe_lif         : 0x8000          psn_fec         : 0x20000410  encap_id  : 0x8000
dest_gport      : 0x6C0000D1      ing_gport       : 0x18908000  egr_gport : 0x18A08000
imp_flow_label : Yes              disp_flow_label : Yes
```



CHAPTER 6

EVPN Virtual Private Wire Service (VPWS) Single Homed

EVPN-VPWS single homed is a BGP control plane solution for point-to-point services. It has the ability to forward traffic from or to one network to another using the Ethernet Segment without MAC lookup.

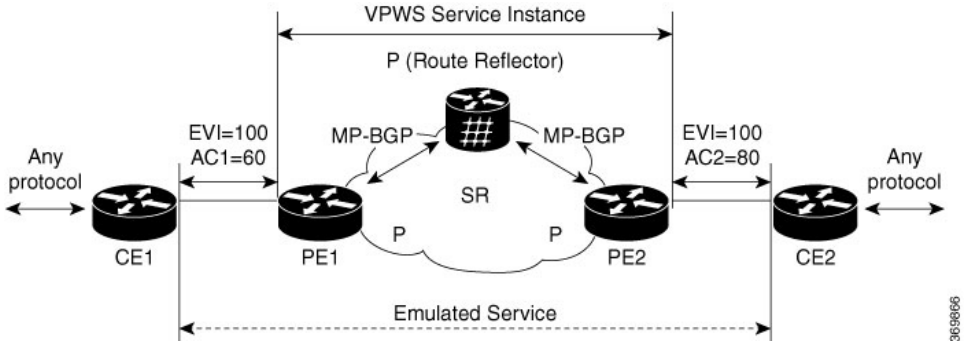
EVPN VPWS single homed technology works on IP and MPLS core. IP core to support BGP and MPLS core for switching packets between the endpoints.

Information About EVPN-VPWS

The EVPN-VPWS solution supports per EVI Ethernet Auto Discovery route. EVPN defines a new BGP Network Layer Reachability Information (NLRI) that is used to carry all EVPN routes. BGP Capabilities Advertisement is used to ensure that two speakers support EVPN NLRI (AFI 25, SAFI 70) as per RFC 4760.

The architecture for EVPN VPWS is that the PEs run Multi-Protocol BGP in control-plane. The following image describes the EVPN-VPWS over SR configuration:

Figure 10: EVPN-VPWS over SR Configuration



Benefits of EVPN-VPWS Single Homed

- Scalability is achieved without signaling pseudowires.
- There is ease of provisioning.
- Pseudowires (PWs) are not used.
- EVPN-VPWS Single Homed leverages BGP best-path selection (optimal forwarding).

Prerequisites for EVPN-VPWS

- Ensure BGP is configured for EVPN SAFI.
- MPLS LDP core is used for MPLS LSP between PE. MPLS LDP core is required when Segement Routing is not used.
- CE-facing interface, such as service instance, is Ethernet family without IP address on PE.
- BGP session between PEs with 'address-family l2vpn evpn' to exchange EVPN routes.
- A BGP Route Reflector is supported.
- IGP, such as ISIS, core for IP reachability between PEs and BGP next-hop reachability.

Restrictions for EVPN-VPWS

- The combination of EVPN ID and VPWS Instance ID must be unique according to ASN.
- Cisco Multiprotocol Label Switching Traffic Engineering (MPLS-TE) core is *not* supported.
- inter-AS Option B is *not* supported.
- NSR is *not* supported for l2vpn family.
- Ensure that Cisco Nonstop Forwarding (NSF) is configured on BGP, OSPF(iBGP), and MPLS.
- NSF is supported, you should see neigh flap, but not traffic drop.
- Without NSF, if you are doing Stateful Switchover (SSO), then you would see traffic drop for l2vpn evpn traffic.
- **evpn vc stats** do *not* work in the **show l2vpn evpn vc id detail** command.
- ELB is *not* supported on EVPN.
- L2VPN traffic is not load balanced for inner payload src-ip, dst-ip, src-dst-ip hashing algorithms in the egress PoCh interface. We recommend you to use other hashing algorithms like src-mac, dst-mac, src-dst-mac.

Scaling Information

4000 EVPN-VPWS service instances are supported.

How to Configure EPVN-VPWS

The following steps are performed to configure EVPN-VPWS

- Configuring BGP for EVPN-VPWS
- Configuring EVPN-VPWS Instance

Configuring BGP for EVPN-VPWS

To configure EVPN-VPWS in BGP, follow these steps:

Procedure

```
router bgp 1
address-family l2vpn evpn
neighbor 192.168.0.1 activate
exit-address-family
```

Configuring EVPN-VPWS Instance

To configure EVPN VPWS instance, follow these steps:

Procedure

```
enable
configure terminal
l2vpn evpn instance 11 point-to-point
vpws context test
service target 100 source 100
member GigabitEthernet0/0/0 service-instance 10
no shut
end
```

Rewrite for EVI Service Instance

You need to have the rewrite command when the VLANs are mismatched on the remote ACs. This allows ingress traffic movement. To configure EVPN-VPWS service instance for rewrite, follow these steps:

Procedure

```
interface GigabitEthernet0/0/1
service instance 2 ethernet
encapsulation dot1q 2
rewrite ingress tag pop 1 symmetric
end
```

Configuring EVPN-VPWS for Logging

To configure EVPN-VPWS for logging, follow these steps:

Procedure

```
enable
configure terminal
l2vpn evpn logging vc-state
end
```

Verfiying EVPN-VPWS Instance

Verifying EVPN-VPWS Configuration

You can verify the configuration using the following show commands:

- **show l2vpn evpn summary**
- **show l2vpn evpn evi (<evpn-id> | all) [detail]**
- **show l2vpn evpn rib ead [detail] |evi**
- **show l2vpn evpn checkpoint**
- **show l2vpn evpn route-target [<rt>]**
- **show bgp l2vpn evpn**
- **show l2vpn evpn memory [detail]**

This command displays a summary of L2VPN EVPN with total number of EVIs, VCs and routes.

```
show l2vpn evpn summary

L2VPN EVPN VPWS:
  EVIs (point-to-point): 1
  Total VCs: 1
    1 up, 0 down, 0 admin-down, 0 hot-standby, 0 other
  Total EVPN EAD routes: 2
    1 local, 1 remote
  Total EVI EAD routes: 2
    1 local, 1 remote (1 in-use)
  BGP: ASN 1, address-family l2vpn evpn configured
  Router ID: 192.168.0.2
```

This command displays brief or detail info for EVIs.

```
show l2vpn evpn evi 100 det

EVPN instance: 100 (point-to-point)
  RD: 192.168.0.2:100 (auto)
  Import-RTs: 1:100
  Export-RTs: 1:100
  Total VCs: 1
    1 up, 0 down, 0 admin-down, 0 hot-standby, 0 other
  Total EAD routes: 2
    1 local, 1 remote (1 in-use)
```

This command displays the contents of the global EVPN route.

```
show l2vpn evpn rib ead

+- Origin of entry                                (i=iBGP/e=eBGP/L=Local)
| +- Best path                                    (Yes/No)?
| |
```

```
v v
O B         RD              Ethernet Segment Id    Eth Tag     Next Hop
-+-+---------------------+-----------------------+--------+---------------
i Y 192.168.0.3:100       0000.0000.0000.0000.0000 2        192.168.0.3
L - 192.168.0.2:100       0000.0000.0000.0000.0000 1
```

```
show l2vpn evpn rib ead evi

+- Origin of entry                                (i=iBGP/e=eBGP/L=Local)
| +- Provisioned                                  (Yes/No)?
| | +- Best path                                  (Yes/No)?
| | |
v v v
O P B  EVI    Ethernet Segment Id    Eth Tag     Next Hop      Label
-+-+-+-----+-----------------------+--------+---------------+--------
i Y Y 100   0000.0000.0000.0000.0000 2        192.168.0.3     16
L - - 100   0000.0000.0000.0000.0000 1                        16
```

```
show l2vpn evpn checkpoint

EVPN Checkpoint info for active RP
Checkpointing is allowed
Bulk-sync checkpointed state for 0 VC
ISSU Context:95, Compatible:1, Negotiated L2HW types: 0
```

This command displays the contents of the global route-target (RT).

```
show l2vpn evpn route-target

Route Target           EVPN Instances
1:100                  100
```

```
show bgp l2vpn evpn

BGP table version is 4, local router ID is 192.168.0.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

     Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 192.168.0.2:100
 *>  [1][192.168.0.2:100][00000000000000000000][5]/23
                        ::                                    32768 ?
Route Distinguisher: 192.168.0.3:100
 *>i [1][192.168.0.3:100][00000000000000000000][6]/23
                        192.168.0.3              0    100      0 ?
```

This command displays brief or detail EVPN memory usage.

```
show l2vpn evpn memory

  Allocator-Name                  In-use/Allocated            Count
  ------------------------------------------------------------------------------
  EVPN DB                   :        648/65632      (  0%) [      9] Chunk
  EVPN EAD DB               :        432/65632      (  0%) [      6] Chunk
  EVPN EAD Handle Table     :      21856/22040      ( 99%) [      2]
  EVPN EAD Paths            :        104/65632      (  0%) [      1] Chunk
  EVPN EAD Routes           :         96/65648      (  0%) [      2] Chunk
  EVPN RIB MGR              :        976/1344       ( 72%) [      4]
  EVPN RIB NHs              :          0/10096      (  0%) [      0] Chunk
  EVPN RIB RTs              :         96/10096      (  0%) [      2] Chunk
  EVPN RIB msg              :          0/10096      (  0%) [      0] Chunk
  EVPN Thread               :       1684/2144       ( 78%) [      5]
  EVPN context chunk        :        768/32864      (  2%) [      1] Chunk
```

```
 EVPN context handle table :      70968/71152      ( 99%) [      2]
 EVPN dtrace elem per-cont :       1280/65632      (  1%) [     20] Chunk
 EVPN dtrace stridx        :    1194876/1194968    ( 99%) [      1]
 EVPN dtrace stridx freeli :     132764/132856     ( 99%) [      1]
 EVPN dtrace stridx hash   :         76/168        ( 45%) [      1]
 EVPN dtrace stridx slots  :     265532/265624     ( 99%) [      1]
 EVPN dtrace stridx2slot   :     132764/132856     ( 99%) [      1]
 EVPN instance chunk       :        168/10096      (  1%) [      1] Chunk
 EVPN rt-db ee             :        124/216        ( 57%) [      1]
 EVPN rt-db rte            :        204/296        ( 68%) [      1]

 Total allocated: 2.121 Mb, 2172 Kb, 2225088 bytes
```

Verifying EVPN-VPWS Configuration for Logging

You can verify the logging using the **show l2vpn evpn vc** command.

This command displays brief information for VCs.

```
show l2vpn evpn vc all

 EVPN ID Source   Target   Type   Name/Interface                   Status
------- -------- -------- ------ -------------------------------- ----------
100     1        2        p2p    vc100                            up
                                 Et0/0                            up
```

This command displays detail information for VCs.

```
show l2vpn evpn vc all detail

 EVPN name: vc100, state: up, type: point-to-point
  EVPN ID: 100
  VPWS Service Instance ID:  Source 1, Target 2
  Labels:  Local 16, Remote 16
  Next Hop Address: 192.168.0.3
  Associated member Et0/0 is up, status is up
  Dataplane:
    SSM segment/switch IDs: 4098/4097 (used), PWID: 1
  Rx Counters
    78 input transit packets, 26425 bytes
    0 drops
  Tx Counters
    79 output transit packets, 28240 bytes
    0 drops
  5 VC FSM state transitions, Last 5 shown
    Prov: Idle -> Prov, Tue Sep 29 13:15:37.848 (00:52:21 ago)
    AdmUp: Prov -> LocWait, Tue Sep 29 13:15:40.287 (00:52:18 ago)
    LocUp: LocWait -> RemWait, Tue Sep 29 13:15:40.287 (00:52:18 ago)
    RemUp: RemWait -> Act, Tue Sep 29 13:17:19.368 (00:50:39 ago)
    DpUp: Act -> Est, Tue Sep 29 13:17:19.371 (00:50:39 ago)
```

Troubleshooting

Virtual Circuit (VC) is in Down state

EVPN VPWS protocol has no communication of VC state between endpoints. Furthemore LDP transport LSP is unidirectional and there is no end-to-end checking for connectivity. VC can be up on one end and down on the other end in the following cases:

- Core-facing mpls dataplane down on one side only. For example, if loopback configured with /24 on one-end and configured correctly with /32 at other end.

- UUT has no remote EVPN EAD route from peer. Several variants:
  - Peer never sent it.
  - Peer sent it, but RT mismatch: No intersection between UUT Import-RT and peer Export-RT.
  - Peer sent it, RT matches, but etag mismatch: For service etags tgt/src, UUT has x/y, peer has y/z.

Problem VC is in down state.

Possible Cause None

Solution Perform these steps to check whether the VC is not active:

Solution

- **Solution** Check if any VC is not active.
- **Solution** Identify EVIs that has not got an active VCs
- **Solution** Gather information for the EVIs that has not got an active VCs
- **Solution** Locate the not active VCs for the EVI
- **Solution** Display detail information of the not active VC

Solution

```
show l2vpn evpn vc all detail
EVPN name: vc100, state: up, type: point-to-point
  EVPN ID: 100
  VPWS Service Instance ID:  Source 1, Target 2
  Labels:  Local 16, Remote 16

// Must have a valid Local Label. If missing, contact support.

// Must have valid Remote Label. If missing, then there is no matching remote route.
Cross-check with BGP: 'show bgp l2vpn evpn [...] detail'.

  Next Hop Address: 192.168.0.3

// Must have valid Next Hop Address. If missing, then there is no matching remote route.
Cross-check with BGP: 'show bgp l2vpn evpn [...] detail'.

  Associated member Et0/0 is up, status is up

// AC must be up. If not up, check why.

  Dataplane:
    SSM segment/switch IDs: 4098/4097 (used), PWID: 1
  Rx Counters
    78 input transit packets, 26425 bytes
    0 drops
  Tx Counters
    79 output transit packets, 28240 bytes
    0 drops
  5 VC FSM state transitions, Last 5 shown
    Prov: Idle -> Prov, Tue Sep 29 13:15:37.848 (00:52:21 ago)
    AdmUp: Prov -> LocWait, Tue Sep 29 13:15:40.287 (00:52:18 ago)
    LocUp: LocWait -> RemWait, Tue Sep 29 13:15:40.287 (00:52:18 ago)
    RemUp: RemWait -> Act, Tue Sep 29 13:17:19.368 (00:50:39 ago)
    DpUp: Act -> Est, Tue Sep 29 13:17:19.371 (00:50:39 ago)
```

```
// Pay close attention to last line of VC FSM history. The format is:
// <Event>: <OldState> -> <NewState>
// Troubleshooting info appears below.
```

VC FSM History

Problem The state of the VC is Prov — Provisioned: VC is disabled.

Possible Cause None

Solution Perform these steps for a solution to the state:

- **Solution** Check BGP is running.
- **Solution** Check BGP 'address-family l2vpn evpn' is configured.
- **Solution** Check VC is not shutdown.

Problem The state of the VC is LocWait — Local-Wait: Waiting for local AC information to come up.

Possible Cause None

Solution Check AC is up.

Problem The state of the VC is Act — Activating: Control plane ok. Trying to activate dataplane.

Possible Cause None

- **Solution** Check core facing information is up.
- **Solution** Check Segment-Routing is configured and preferred.

Remote-Wait State

Problem The state of the VC is RemWait — Remote-Wait: Waiting for matching remote route.

Possible Cause This state occurs due to no matching remote route for the VC. A matching remote route means all of the following are true:

- Route is present in BGP. Requires a local EVI to have route target in the route.
- Remote path is best path.
- Route is present in global EVPN route.
- Route is present in EVI route. Requires the EVI to have route target in the route.
- Route has ETag which matches the VC source identity. (**service target <tgt-id> source <src-id>**).

Solution Perform these steps to check whether the VC is in remote wait state:

Solution

- Check for EVI configuration mismatch.
- Check for VC configuration mismatch.
- Check if the remote route is present in BGP.
    - If no remote route then check if
        - remote route was discarded by BGP due to RT filter
        - peer did not send route to UUT
        - EVI or VC configuration mismatch
        - all the prerequisites are satisfied

- If a remote route is present in global EVPN then check if the remote route is present in EVI route.
  - **Solution** Check for EVI or VC configuration mismatch.

Configuration Examples for EVPN-VPWS Instance

The following example is for configuration for an EVPN-VPWS instance.

Example: EVPN-VPWS Instance Configuration

```
Router(config)#l2vpn evpn instance 11 point-to-point
Router(config-evpn-evi)#rd 1:1
Router(config-evpn-evi)#vpws context test
Router(config-evpn-vpws)#service target 100 source 100
Router(config-evpn-vpws)#member GigabitEthernet0/0/0 service-instance 10
Router(config-evpn-vpws)#no shut
```

The following example has running configurations on PE1 and PE2

Example: EVPN-VPWS PE1 configuration

```
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
 ip ospf 1 area 0
!
interface GigabitEthernet0/0/0
 description CE1 facing
 no ip address
!
service instance 300 ethernet
   encapsulation dot1q 300
   rewrite ingress tag pop 1 symmetric

 l2vpn evpn instance 100 point-to-point
 !
 vpws context vc100
 service target 2 source 1
 member GigabitEthernet0/0/0 service-instance 300
!
interface GigabitEthernet0/0/1
 description Core facing
 ip address 10.0.1.1 255.255.255.0
 ip ospf 1 area 0
 mpls ip
!
router ospf 1
 router-id 10.1.1.1
!
router bgp 1
 bgp router-id 10.1.1.1
 neighbor 2.2.2.2 remote-as 1
 neighbor 2.2.2.2 update-source Loopback0
 !
 address-family ipv4
  neighbor 2.2.2.2 activate
 exit-address-family
 !
```

```
 address-family l2vpn evpn
  neighbor 2.2.2.2 activate
 exit-address-family
!
l2vpn evpn instance 100 point-to-point
 !
 vpws context vc100
  service target 2 source 1
  member GigabitEthernet0/0/0
!
mpls ldp router-id Loopback0
!
```

Example: EVPN-VPWS PE2 configuration

```
interface Loopback0
 ip address 2.2.2.2 255.255.255.255
 ip ospf 1 area 0
!
interface GigabitEthernet0/0/0
 description CE2 facing
 no ip address
!
service instance 300 ethernet
   encapsulation dot1q 300
   rewrite ingress tag pop 1 symmetric

 l2vpn evpn instance 100 point-to-point
 !
 vpws context vc100
 service target 2 source 1
 member GigabitEthernet0/0/0 service-instance 300

interface GigabitEthernet0/0/1
 description Core facing
 ip address 10.0.1.2 255.255.255.0
 ip ospf 1 area 0
 mpls ip
!
router ospf 1
 router-id 2.2.2.2
!
router bgp 1
 bgp router-id 2.2.2.2
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Loopback0
 !
 address-family ipv4
  neighbor 10.1.1.1 activate
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor 10.1.1.1 activate
 exit-address-family
!
l2vpn evpn instance 100 point-to-point
 !
 vpws context vc100
  service target 1 source 2
  member GigabitEthernet0/0/0
!
```

```
mpls ldp router-id Loopback0
!
```

Additional References for EVPN-VPWS

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Standards and RFCs

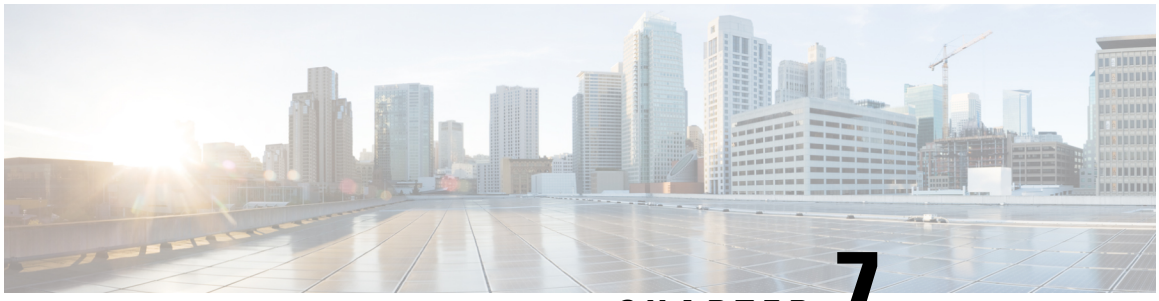
Standard/RFC	Title
RFC 7432	*BGP MPLS-Based Ethernet VPN*
Standard	*VPWS support in EVPN*

MIBs

MIB	MIBs Link
•	—

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 7

VPLS MAC Address Withdrawal

The VPLS MAC Address Withdrawal feature provides faster convergence by removing (or unlearning) MAC addresses that have been dynamically learned. A Label Distribution Protocol (LDP)-based MAC address withdrawal message is used for this purpose. A MAC list Type Length Value (TLV) is part of the MAC address withdrawal message. No configuration is needed.

Information About VPLS MAC Address Withdrawal

VPLS MAC Address Withdrawal

The VPLS MAC Address Withdrawal feature provides faster convergence by removing (or unlearning) MAC addresses that have been dynamically learned. A Label Distribution Protocol (LDP)-based MAC address withdrawal message is used for this purpose. A MAC list Type Length Value (TLV) is part of the MAC address withdrawal message.

The **debug mpls ldp messages** and **debug mpls ldp session io** commands support monitoring of MAC address withdrawal messages being exchanged between LDP peers. Any Transport over Multiprotocol Label Switching (AToM) might provide other means to display or monitor MAC address withdrawal messages. The Tag Distribution Protocol (TDP) is not supported because AToM uses only LDP for the MAC address withdrawal message.

PE devices learn the remote MAC addresses and directly attached MAC addresses on customer-facing ports by deriving the topology and forwarding information from packets originating at customer sites. To display the number of MAC address withdrawal messages, enter the **show mpls l2transport vc detail** command, as shown in the following example:

```
Device# show mpls l2transport vc detail

Local interface: VFI TEST VFI up
  MPLS VC type is VFI, interworking type is Ethernet
  Destination address: 10.1.1.1, VC ID: 1000, VC status: up
    Output interface: Se2/0, imposed label stack {17}
    Preferred path: not configured
    Default path: active
    Next hop: point2point
  Create time: 00:04:34, last status change time: 00:04:15
```

```
  Signaling protocol: LDP, peer 10.1.1.1:0 up
    Targeted Hello: 10.1.1.1(LDP Id) -> 10.1.1.1
    MPLS VC labels: local 16, remote 17
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
    MAC Withdraw: sent 5, received 3
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 0, send 0
    byte totals:   receive 0, send 0
    packet drops:  receive 0, send 0
```

Note The MAC Address feature is enabled by default. Some show commands may not display the MAC withdraw counters in the command output. This does not indicate that the feature is disabled.

VPLS MAC Address Withdrawal Using Commands Associated with L2VPN Protocol-Based Feature

The VPLS MAC Address Withdrawal feature provides faster convergence by removing (or unlearning) MAC addresses that have been dynamically learned. A Label Distribution Protocol (LDP)-based MAC address withdrawal message is used for this purpose. A MAC list Type Length Value (TLV) is part of the MAC address withdrawal message.

The **debug mpls ldp messages** and **debug mpls ldp session io** commands support monitoring of MAC address withdrawal messages being exchanged between LDP peers. Any Transport over Multiprotocol Label Switching (AToM) might provide other means to display or monitor MAC address withdrawal messages. The Tag Distribution Protocol (TDP) is not supported because AToM uses only LDP for the MAC address withdrawal message.

PE devices learn the remote MAC addresses and directly attached MAC addresses on customer-facing ports by deriving the topology and forwarding information from packets originating at customer sites. To display the number of MAC address withdrawal messages, enter the **show l2vpn atom vc detail** command, as shown in the following example:

```
Device# show l2vpn atom vc detail

Local interface: VFI TEST VFI up
  MPLS VC type is VFI, interworking type is Ethernet
  Destination address: 10.1.1.1, VC ID: 1000, VC status: up
    Output interface: Se2/0, imposed label stack {17}
    Preferred path: not configured
    Default path: active
    Next hop: point2point
  Create time: 00:04:34, last status change time: 00:04:15
  Signaling protocol: LDP, peer 10.1.1.1:0 up
    Targeted Hello: 10.1.1.1(LDP Id) -> 10.1.1.1
    MPLS VC labels: local 16, remote 17
    Group ID: local 0, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
    MAC Withdraw: sent 5, received 3
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 0, send 0
```

```
byte totals:   receive 0, send 0
packet drops:  receive 0, send 0
```

How MAC Address Withdrawal Works with H-VPLS N-PE Redundancy with MPLS Access

If the pseudowire between the user provider edge (U-PE) device and network provider edge (N-PE) device fails, the L2VPN Pseudowire Redundancy feature on the U-PE device activates the standby pseudowire. In addition, the U-PE device sends a Label Distribution Protocol (LDP) MAC address withdrawal request to the new N-PE device, which forwards the message to all pseudowires in the virtual private LAN service (VPLS) core and flushes its MAC address table.

If a switched virtual interface (SVI)bridge domain interface (BDI) on the N-PE device fails, the L2VPN Pseudowire Redundancy feature activates the standby pseudowire and the U-PE device sends a MAC withdrawal message to the newly active N-PE device.

How MAC Address Withdrawal Works with H-VPLS N-PE Redundancy with QinQ Access

If a failure occurs in the customer-switched network, a spanning-tree Topology Change Notification (TCN) is issued to the network provider edge (N-PE) device, which issues a Label Distribution Protocol (LDP)-based MAC address withdrawal message to the peer N-PE devices and flushes its MAC address table.

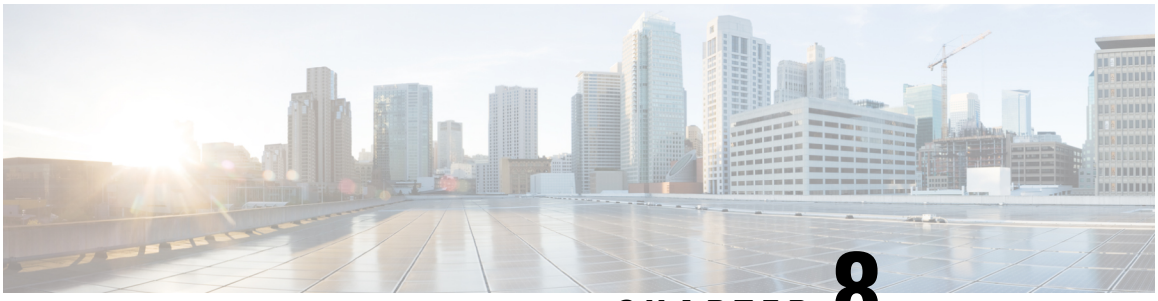
Additional References for Any Transport over MPLS

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Cisco IOS Multiprotocol Label Switching Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 8

H-VPLS N-PE Redundancy for MPLS Access

The H-VPLS N-PE Redundancy for MPLS Access feature enables two network provider edge (N-PE) devices to provide failover services to a user provider edge (U-PE) device in a hierarchical virtual private LAN service (H-VPLS). Having redundant N-PE devices provides improved stability and reliability against link and node failures.

Prerequisites for H-VPLS N-PE Redundancy for MPLS Access

- Before configuring this feature, configure your hierarchical virtual private LAN service (H-VPLS) network and make sure it is operating correctly.
- Make sure that the PE-to-customer edge (CE) interface is configured with a list of allowed VLANs.
- To provide faster convergence, you can enable the MPLS Traffic Engineering—Fast Reroute feature in the Multiprotocol Label Switching (MPLS) core.
- Enable the L2VPN Pseudowire Redundancy feature on the user provider edge (U-PE) devices for MPLS access.

Restrictions for H-VPLS N-PE Redundancy for MPLS Access

- This feature cannot be used with the VPLS Autodiscovery feature on pseudowires that attach to user provider edge (U-PE) devices. When you create the virtual private LAN service (VPLS), you can manually create the virtual forwarding interface (VFI).
- You cannot configure more than one pseudowire to carry the bridge protocol data unit (BPDU) information between the network provider edge (N-PE) devices.

- You cannot configure a local loopback address as a neighbor when you configure the H-VPLS N-PE Redundancy feature on N-PE devices.
- Only two N-PE devices can be connected to each U-PE device.

Information About H-VPLS N-PE Redundancy for MPLS Access

How H-VPLS N-PE Redundancy for MPLS Access

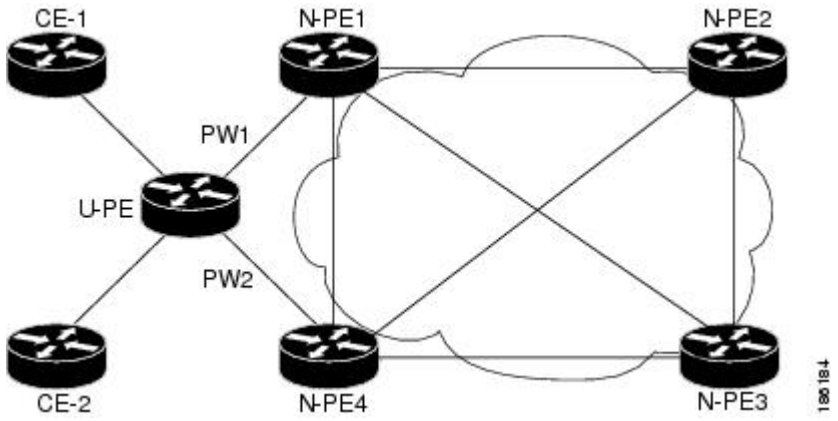
In a network configured with the H-VPLS N-PE Redundancy feature, the user provider edge (U-PE) device is connected to two network provider edge (N-PE) devices. This feature provides a level of redundancy that can tolerate both link and device faults. If a failure occurs in the network that disables one N-PE device from transmitting data, the other N-PE device takes over.

H-VPLS N-PE Redundancy with MPLS Access Based on Pseudowire Redundancy

For the H-VPLS Redundancy with MPLS Access feature based on pseudowire redundancy, the Multiprotocol Label Switching (MPLS) network has pseudowires to the virtual private LAN service (VPLS) core network provider edge (N-PE) devices.

As shown in the figure below, one pseudowire transports data between the user provider edge (U-PE) device and its peer N-PE devices. When a failure occurs along the path of the U-PE device, the backup pseudowire and the redundant N-PE device become active and start transporting data.

Figure 11: H-VPLS N-PE Redundancy for MPLS Access Based on Pseudowire Redundancy



How to Configure H-VPLS N-PE Redundancy for MPLS Access

Configuring the VPLS Pseudowire Between the N-PE Devices

Configuring network provider edge (N-PE) redundancy in a hierarchical Virtual Private LAN service (H-VPLS) network requires that you define the VPLS pseudowire for transporting bridge protocol data unit (BPDU) packets (described here) and that you connect that pseudowire to the native VLAN (described in the next task). This configuration provides a redundancy that provides improved reliability against link and node failures.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *name* **manual** **Example:** `Device(config)# l2 vfi vfitest1 manual`	Creates a Layer 2 virtual forwarding interface (VFI) and enters Layer 2 VFI manual configuration mode.
Step 4	**vpn id** *id-number* **Example:** `Device(config-vfi)# vpn id 10`	Specifies the VPN ID.
Step 5	**bridge-domain** *bridge-id*	Configures the router to derive bridge domains from the encapsulation VLAN list.
Step 6	**forward permit l2protocol all** **Example:** `Device(config-vfi)# forward permit l2protocol all`	Creates a pseudowire that is to be used to transport BPDU packets between the two N-PE devices.
Step 7	**neighbor** *remote-router-id vc-id* {**encapsulation** *encapsulation-type* | **pw-class** *pw-name*} [**no-split-horizon**] **Example:**	Specifies the peer IP address of the redundant N-PE device and the type of tunnel signaling and encapsulation mechanism.

	Command or Action	Purpose
	`Device(config-vfi)# neighbor 10.2.2.2 3 encapsulation mpls`	
Step 8	**end** **Example:** `Device(config-vfi)# end`	Exits Layer 2 VFI manual configuration mode and returns to privileged EXEC mode.

Example

You can also configure the VPLS pseudowire between the N-PE devices using this alternate method.

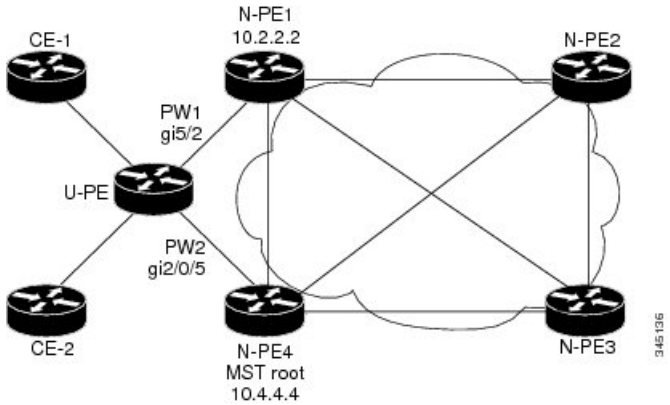
```
RoutDeviceer> enable
Device# configure terminal
Device(config)# l2vpn vfi context vfi110
Device(config-vfi)# vpn id 10
Device(config-vfi)# exit
Device(config)# bridge-domain 100
Device(config-bdomain)# member vfi vfi110
Device(config-vfi)# member 172.16.10.2 4 encapsulation mpls
Device(config-vfi)# end
```

Configuration Examples for H-VPLS N-PE Redundancy for MPLS Access

Example: H-VPLS N-PE Redundancy for MPLS Access

The figure below shows a configuration that is set up for the H-VPLS N-PE Redundancy with MPLS Access feature.

Figure 12: H-VPLS N-PE Redundancy with MPLS Access Topology



The table below shows the configuration of two network provider edge (N-PE) devices.

Table 6: Example: H-VPLS N-PE Redundancy for MPLS Access

N-PE1	N-PE4
l2 vfi l2trunk manual vpn id 10 bridge-domain 10 forward permit l2protocol all neighbor 10.4.4.4 encapsulation mpls ! interface Vlan1 no ip address xconnect vfi l2trunk ! spanning-tree mode mst spanning-tree extend system-id ! spanning-tree mst configuration revision 10 instance 1 vlan 20 ! interface GigabitEthernet5/2 switchport switchport trunk encapsulation dot1q switchport trunk allowed vlan 20 switchport mode trunk interface GigabitEthernet 0/5/2 service instance 5 ethernet encapsulation dot1q 10 bridge-domain 10	l2 vfi l2trunk manual vpn id 10 bridge-domain 10 forward permit l2protocol all neighbor 10.2.2.2 encapsulation mpls ! interface Vlan1 no ip address xconnect vfi l2trunk ! spanning-tree mode mst spanning-tree extend system-id ! spanning-tree mst configuration revision 10 instance 1 vlan 20 ! spanning-tree mst 1 priority 0 ! interface GigabitEthernet2/0/5 switchport switchport trunk allowed vlan 20 switchport mode trunk mls qos trust dscp interface GigabitEthernet 0/5/2 service instance 5 ethernet encapsulation dot1q 10 bridge-domain 10

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Cisco IOS Multiprotocol Label Switching Command Reference
L2VPN pseudowire redundancy	“L2VPN Pseudowire Redundancy” feature module in the *MPLS Layer 2 VPNs Configuration Guide*.
H-VPLS	“ Configuring VPLS ” in the “Configuring Multiprotocol Label Switching on the Optical Services Modules” chapter in the *Optical Services Modules Installation and Configuration Notes*, 12.2SR document.
MPLS traffic engineering	“MPLS Traffic Engineering Fast Reroute Link and Node Protection” feature module in the *MPLS Traffic Engineering: Path, Link, and Node Protection Configuration Guide* (part of the Multiprotocol Label Switching Configuration Guide Library)

Standards

Standard	Title
http://www.ietf.org/rfc/rfc4447.txt	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*
http://www3.ietf.org/proceedings/06mar/IDs/draft-ietf-l2vpn-vpls-ldp-08.txt	*Virtual Private LAN Services over MPLS*
http://www.ietf.org/internet-drafts/draft-ietf-pwe3-segmented-pw-02.txt	*Segmented Pseudo Wire*
draft-ietf-pwe3-vccv-10.txt	*Pseudo Wire Virtual Circuit Connectivity Verification (VCCV)*
draft-ietf-pwe3-oam-msg-map-03.txt	*Pseudo Wire (PW) OAM Message Mapping*

MIBs

MIB	MIBs Link
Pseudowire Emulation Edge-to-Edge MIBs for Ethernet, Frame Relay, and ATM Services	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Glossary

CE device—customer edge device. A device that belongs to a customer network, which connects to a PE device to utilize MPLS VPN network services.

LAN—local-area network. High-speed, low-error data network covering a relatively small geographic area. LANs connect workstations, peripherals, terminals, and other devices in a single building or other geographically limited areas.

MPLS—Multiprotocol Label Switching. A packet-forwarding technology, used in the network core, that applies data link layer labels to tell switching nodes how to forward data, resulting in faster and more scalable forwarding than network layer routing normally can do.

MSTP—Multiple Spanning Tree Protocol. MSTP enables multiple VLANs to be mapped to the same spanning-tree instance, reducing the number of spanning-tree instances needed to support a large number of VLANs.

N-PE—network provider edge device. This device acts as a gateway between the MPLS core and edge domains.

PE device—provider edge device. The PE device is the entry point into the service provider network. The PE device is typically deployed on the edge of the network and is administered by the service provider.

pseudowire—A pseudowire is a virtual connection that, in the context of VPLS, connects two SVIs. It is a mechanism that carries the elements of an emulated service from one PE device to one or more PE devices over a packet switched network (PSN). A pseudowire is bidirectional and consists of a pair of unidirectional MPLS virtual circuits (VCs). A pseudowire can be used to connect a point-to-point circuit.

QinQ—An IEEE 802.1Q VLAN tunnel. A mechanism for constructing multipoint Layer 2 VPN using Ethernet switches.

redundancy—The duplication of devices, services, or connections so that, in the event of a failure, they can perform the work of those that failed.

router—A network layer device that uses one or more metrics to determine the optimal path along which network traffic should be forwarded. Routers forward packets from one network to another based on network layer information.

spanning tree—Loop-free subset of a network topology.

U-PE—user provider edge device. This device connects CE devices to the service.

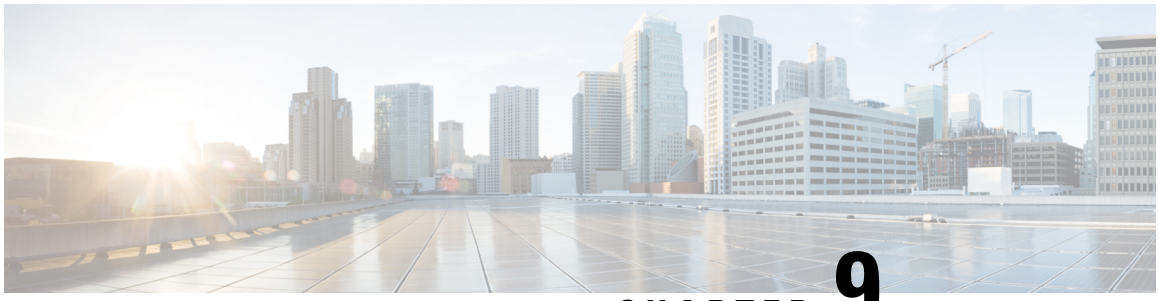
VFI—virtual forwarding instance. A VFI is a collection of data structures used by the data plane, software-based or hardware-based, to forward packets to one or more VCs.

VLAN—Virtual LAN. Group of devices on one or more LANs that are configured (using management software) 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.

VPLS—Virtual Private LAN Service. VPLS describes an architecture that delivers Layer 2 service that emulates an Ethernet LAN across a wide-area network (WAN) and inherits the scaling characteristics of a LAN.

VPLS redundancy—Also called N-PE redundancy. Allows U-PEs to be dual-honed (to their N-PEs) in a loop-free topology with MPLS or QinQ as the access or aggregation domain.

VPN—Virtual Private Network. Allows IP traffic to travel securely over public TCP/IP networks and the Internet by encapsulating and encrypting all IP packets. VPN uses a tunnel to encrypt all information at the IP level.



CHAPTER 9

VPLS Autodiscovery BGP Based

VPLS Autodiscovery enables Virtual Private LAN Service (VPLS) provider edge (PE) devices to discover other PE devices that are part of the same VPLS domain. VPLS Autodiscovery also automatically detects when PE devices are added to or removed from a VPLS domain. As a result, with VPLS Autodiscovery enabled, you no longer need to manually configure a VPLS domain and maintain the configuration when a PE device is added or deleted. VPLS Autodiscovery uses the Border Gateway Protocol (BGP) to discover VPLS members and set up and tear down pseudowires in a VPLS domain.

This module describes how to configure BGP-based VPLS Autodiscovery.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Restrictions for VPLS Autodiscovery BGP Based

- Virtual Private LAN Service (VPLS) Autodiscovery supports only IPv4 addresses.
- VPLS Autodiscovery uses Forwarding Equivalence Class (FEC) 129 to convey endpoint information. Manually configured pseudowires use FEC 128.
- VPLS Autodiscovery is not supported with Layer 2 Tunnel Protocol Version 3 (L2TPv3).
- You can configure both autodiscovered and manually configured pseudowires in a single virtual forwarding instance (VFI). However, you cannot configure different pseudowires on the same peer PE device.

- After enabling VPLS Autodiscovery, if you manually configure a neighbor by using the **neighbor** command and both peers are in autodiscovery mode, each peer will receive discovery data for that VPLS. To prevent peers from receiving data for the VPLS domain, manually configure route target (RT) values.
- If you manually configure multiple pseudowires and target different IP addresses on the same PE device for each pseudowire, do not use the same virtual circuit (VC) ID to identify pseudowires that terminate at the same PE device.
- If you manually configure a neighbor on one PE device, you cannot configure the same pseudowire in the other direction by using autodiscovery on another PE device.
- Tunnel selection is not supported with autodiscovered neighbors.
- Up to 16 RTs are supported per VFI.
- The same RT is not allowed in multiple VFIs on the same PE device.
- The Border Gateway Protocol (BGP) autodiscovery process does not support dynamic, hierarchical VPLS. User-facing PE (U-PE) devices cannot discover network-facing PE (N-PE) devices, and N-PE devices cannot discover U-PE devices.
- Pseudowires for autodiscovered neighbors have split horizon enabled. (A split horizon is enabled by default on all interfaces. A split horizon blocks route information from being advertised by a device, irrespective of the interface from which the information originates.) Therefore, manually configure pseudowires for hierarchical VPLS. Ensure that U-PE devices do not participate in BGP autodiscovery for these pseudowires.
- Do not disable split horizon on autodiscovered neighbors. Split horizon is required with VPLS Autodiscovery.
- The provisioned peer address must be a /32 address bound to the peer's Label Distribution Protocol (LDP) router ID.
- A peer PE device must be able to access the IP address that is used as the local LDP router ID. Even if the IP address is not used in the **xconnect** command on the peer PE device, the IP address must be reachable.

Information About VPLS Autodiscovery BGP Based

How VPLS Works

Virtual Private LAN Service (VPLS) allows Multiprotocol Label Switching (MPLS) networks to provide multipoint Ethernet LAN services, also known as Transparent LAN Services (TLS). All customer sites in a VPLS appear to be on the same LAN, even though these sites might be in different geographic locations.

How the VPLS Autodiscovery BGP Based Feature Works

VPLS Autodiscovery enables each Virtual Private LAN Service (VPLS) provider edge (PE) device to discover other PE devices that are part of the same VPLS domain. VPLS Autodiscovery also tracks PE devices when they are added to or removed from a VPLS domain. Autodiscovery and signaling functions use the Border Gateway Protocol (BGP) to find and track PE devices.

BGP uses the Layer 2 VPN (L2VPN) Routing Information Base (RIB) to store endpoint provisioning information, which is updated each time any Layer 2 virtual forwarding instance (VFI) is configured. The prefix and path information is stored in the L2VPN database, which allows BGP to make decisions about the best path. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, this endpoint information is used to configure a pseudowire mesh to support L2VPN-based services.

The BGP autodiscovery mechanism facilitates the configuration of L2VPN services, which are an integral part of the VPLS feature. VPLS enables flexibility in deploying services by connecting geographically dispersed sites as a large LAN over high-speed Ethernet in a robust and scalable IP Multiprotocol Label Switching (MPLS) network. For more information about BGP and the L2VPN address family in relation to VPLS Autodiscovery, see the following chapters in the *IP Routing: BGP Configuration Guide*:

- "L2VPN Address Family" section in the "Cisco BGP Overview" chapter

  "L2VPN Address Family" section in the " Cisco BGP Overview " chapter

- "BGP Support for the L2VPN Address Family" chapter

How Enabling VPLS Autodiscovery Differs from Manually Configuring VPLS

With VPLS Autodiscovery enabled, you no longer need to manually set up Virtual Private LAN Service (VPLS). The commands that you use to set up VPLS Autodiscovery are similar to those that you use to manually configure VPLS, as shown in the table below. VPLS Autodiscovery uses **neighbor** commands in L2VPN address family mode to distribute endpoint information to configure a pseudowire.

Table 7: Manual VPLS Configuration Versus VPLS Autodiscovery Configuration

Manual Configuration of VPLS	VPLS Autodiscovery BGP Based

`l2 vfi vpls1 manual`
` vpn id 100`
` neighbor 10.10.10.1 encapsulation mpls`
` neighbor 10.10.10.0 encapsulation mpls`
` exit`
	`l2 vfi vpls1 autodiscovery`
	` vpn id 100`
	` exit`
	`router bgp 1`
	` no bgp default ipv4-unicast`
	` bgp log-neighbor-changes`
	` bgp update-delay 1`
	` neighbor 10.1.1.2 remote-as 1`
	` neighbor 10.1.1.2 update-source Loopback1`
	` .`
	` .`
	` .`
	` address-family l2vpn vpls`
	` neighbor 10.1.1.2 activate`
	` neighbor 10.1.1.2 send-community extended`
	`exit-address-family`

Configure VPLS Autodiscovery by using the **l2 vfi autodiscovery** command. This command allows a virtual forwarding instance (VFI) to learn and advertise pseudowire endpoints. As a result, you no longer need to enter the **neighbor** command in L2 VFI configuration mode.

However, the **neighbor** command is still supported with VPLS Autodiscovery in L2 VFI configuration mode. You can use the **neighbor** command to allow PE devices that do not participate in the autodiscovery process to join the VPLS domain. You can also use the **neighbor** command with PE devices that have been configured using the Tunnel Selection feature. In addition, you can use the **neighbor** command in hierarchical VPLS configurations that have user-facing PE (U-PE) devices that do not participate in the autodiscovery process and have split-horizon forwarding disabled.

How Enabling VPLS Autodiscovery Differs from Manually Configuring VPLS using the commands associated with the L2VPN Protocol-Based CLIs feature

With VPLS Autodiscovery enabled, you no longer need to manually set up Virtual Private LAN Service (VPLS). The commands that you use to set up VPLS Autodiscovery are similar to those that you use to manually configure VPLS, as shown in the table below. VPLS Autodiscovery uses **neighbor** commands in L2VPN address family mode to distribute endpoint information to configure a pseudowire.

Table 8: Manual VPLS Configuration Versus VPLS Autodiscovery Configuration

Manual Configuration of VPLS	VPLS Autodiscovery BGP Based

`l2vpn vfi context vpls1`
` vpn id 100`
` neighbor 10.10.10.1 encapsulation mpls`
` neighbor 10.10.10.0 encapsulation mpls`
` exit`
	`l2vpn vfi context vpls1`
	` vpn id 100`
	` autodiscovery bgp signaling ldp`
	` exit`
	`router bgp 1`
	` no bgp default ipv4-unicast`
	` bgp log-neighbor-changes`
	` bgp update-delay 1`
	` neighbor 10.1.1.2 remote-as 1`
	` neighbor 10.1.1.2 update-source Loopback1`
	` .`
	` .`
	` .`
	` address-family l2vpn vpls`
	` neighbor 10.1.1.2 activate`
	` neighbor 10.1.1.2 send-community extended`
	`exit-address-family`

Configure VPLS Autodiscovery by using the **autodiscovery** command. This command allows a virtual forwarding instance (VFI) to learn and advertise pseudowire endpoints. As a result, you no longer need to enter the **neighbor** command in L2 VFI configuration mode.

However, the **neighbor** command is still supported with VPLS Autodiscovery in L2 VFI configuration mode. You can use the **neighbor** command to allow PE devices that do not participate in the autodiscovery process to join the VPLS domain. You can also use the **neighbor** command with PE devices that have been configured using the Tunnel Selection feature. In addition, you can use the **neighbor** command in hierarchical VPLS configurations that have user-facing PE (U-PE) devices that do not participate in the autodiscovery process and have split-horizon forwarding disabled.

show Commands Affected by VPLS Autodiscovery BGP Based

The following **show** commands were enhanced for VPLS Autodiscovery:

- The **show mpls l2transport vc detail** command was updated to include Forwarding Equivalence Class (FEC) 129 signaling information for autodiscovered Virtual Private LAN Service (VPLS) pseudowires.
- The **show vfi** command was enhanced to display information related to autodiscovered virtual forwarding instances (VFIs). The new output includes the VPLS ID, the route distinguisher (RD), the route target (RT), and router IDs of discovered peers.
- The **show xconnect** command was updated with the **rib** keyword to provide Routing Information Base (RIB) information about pseudowires.

BGP VPLS Autodiscovery Support on a Route Reflector

By default, routes received from an internal BGP (iBGP) peer are not sent to another iBGP peer unless a full mesh configuration is formed between all BGP devices within an autonomous system (AS). This results in scalability issues. Using Border Gateway Protocol (BGP) route reflectors leads to much higher levels of scalability. Configuring a route reflector allows a device to advertise or reflect the iBGP learned routes to other iBGP speakers.

Virtual Private LAN Service (VPLS) Autodiscovery supports BGP route reflectors. A BGP route reflector can be used to reflect BGP VPLS prefixes without VPLS being explicitly configured on the route reflector.

A route reflector does not participate in autodiscovery; that is, no pseudowires are set up between the route reflector and the PE devices. A route reflector reflects VPLS prefixes to other PE devices so that these PE devices do not need to have a full mesh of BGP sessions. The network administrator configures only the BGP VPLS address family on a route reflector. For an example configuration of VPLS Autodiscovery support on a route reflector, see the “Example: BGP VPLS Autodiscovery Support on Route Reflector” section.

N-PE Access to VPLS Using MST

When a Virtual Private LAN Service (VPLS) network uses multihoming (network-facing PE [N-PE] VPLS redundancy) to prevent a single point of failure of an N-PE device, a bridging loop is introduced. One of the N-PE devices can be set as a Multiple Spanning Tree (MST) root to break the loop. In most cases, the two N-PE devices are also separated by a distance that makes direct physical link impossible. You can configure a virtual link (usually through the same VPLS core network) between the two N-PE devices to pass an MST bridge protocol data unit (BPDU) for path calculation, break the loop, and maintain convergence. The virtual link is created using a special pseudowire between the active and redundant N-PE devices.

While setting up an MST topology for a VPLS PE device, ensure the following:

- The **spanning-tree mode mst** command is enabled on all PE devices (N-PE and user-facing PE [U-PE]) participating in the MST topology.
- A special pseudowire is configured between the two N-PE devices, and these two devices are in the up state.
- The special pseudowire is a manually created virtual forwarding instance (VFI).
- The configuration (inlcuding the MST instance, the Ethernet virtual circuit [EVC], and the VLAN) on all PE devices is the same.
- One of the N-PE devices, and not one of the U-PE devices, is the root for the MST instance.
- The name and revision for the MST configuration are configured to synchronize with the standby Route Processor (RP).

How to Configure VPLS Autodiscovery BGP Based

Enabling VPLS Autodiscovery BGP Based

Note For more information, see Configuring Virtual Private Lan Services.

Perform this task to enable Virtual Private LAN Service (VPLS) PE devices to discover other PE devices that are part of the same VPLS domain.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **autodiscovery** **Example:** `Device(config)# l2 vfi vpls1 autodiscovery`	Enables VPLS Autodiscovery on a PE device and enters L2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**end** **Example:** `Device(config-vfi)# end`	Exits L2 VFI configuration mode and returns to privileged EXEC mode. • Commands take effect after the device exits L2 VFI configuration mode.

Enabling VPLS Autodiscovery BGP Based using the commands associated with the L2VPN Protocol-Based CLIs feature

Perform this task to enable Virtual Private LAN Service (VPLS) PE devices to discover other PE devices that are part of the same VPLS domain.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an L2VPN VFI context and enters L2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**autodiscovery bgp signaling** {**ldp** | **bgp**} **Example:** `Device(config-vfi)# autodiscovery bgp signaling ldp`	Enables the VPLS Autodiscovery: BGP Based feature on the PE device.
Step 6	**end** **Example:** `Device(config-vfi)# end`	Exits L2 VFI configuration mode and returns to privileged EXEC mode. • Commands take effect after the device exits L2 VFI configuration mode.

Configuring BGP to Enable VPLS Autodiscovery

The Border Gateway Protocol (BGP) Layer 2 VPN (L2VPN) address family supports a separate L2VPN Routing Information Base (RIB) that contains endpoint provisioning information for Virtual Private LAN Service (VPLS) Autodiscovery. BGP learns the endpoint provisioning information from the L2VPN database, which is updated each time a Layer 2 virtual forwarding instance (VFI) is configured. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, the endpoint information is used to configure a pseudowire mesh to support L2VPN-based services.

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured using the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast** router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 10.10.10.1 remote-as 65000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local device. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor.

	Command or Action	Purpose
		• In this example, the neighbor at 10.10.10.1 is an internal BGP neighbor.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 10.10.10.1 update-source loopback1`	(Optional) Configures a device to select a specific source or interface to receive routing table updates. • This example uses a loopback interface. The advantage of this configuration is that the loopback interface is not affected by the effects of a flapping interface.
Step 8	Repeat Steps 6 and 7 to configure other BGP neighbors.	—
Step 9	**address-family l2vpn** [**vpls**] **Example:** `Device(config-router)# address-family l2vpn vpls`	Specifies the L2VPN address family and enters address family configuration mode. • The optional **vpls** keyword specifies that the VPLS endpoint provisioning information is to be distributed to BGP peers. • In this example, an L2VPN VPLS address family session is created.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 10.10.10.1 activate`	Enables the exchange of information with a BGP neighbor.
Step 11	**neighbor** {*ip-address* | *peer-group-name*} **send-community** {**both** | **standard** | **extended**} **Example:** `Device(config-router-af)# neighbor 10.10.10.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • In this example, an extended communities attribute is sent to the neighbor at 10.10.10.1.
Step 12	Repeat Steps 10 and 11 to activate other BGP neighbors under an L2VPN address family.	—
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 15	**show vfi** **Example:** `Device# show vfi`	Displays information about the configured VFI instances.
Step 16	**show ip bgp l2vpn vpls** {**all** | **rd** *route-distinguisher*} **Example:** `Device# show ip bgp l2vpn vpls all`	Displays information about the L2VPN VPLS address family.

Configuring BGP to Enable VPLS Autodiscovery using the commands associated with the L2VPN Protocol-Based CLIs feature

The BGP L2VPN address family supports a separate L2VPN Routing Information Base (RIB) that contains endpoint provisioning information for Virtual Private LAN Service (VPLS) Autodiscovery. BGP learns the endpoint provisioning information from the L2VPN database, which is updated each time a Layer 2 virtual forwarding instance (VFI) is configured. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, the endpoint information is used to configure a pseudowire mesh to support L2VPN-based services.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65000`	Enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:**	Disables the IPv4 unicast address family for the BGP routing process.

	Command or Action	Purpose
	`Device(config-router)# no bgp default ipv4-unicast`	**Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured using the **neighbor remote-as** router configuration command unless you configure the **no bgp default ipv4-unicast** router configuration command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 10.10.10.1 remote-as 65000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local device. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor. • In this example, the neighbor at 10.10.10.1 is an internal BGP neighbor.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 10.10.10.1 update-source loopback1`	(Optional) Configures a device to select a specific source or interface to receive routing table updates. • This example uses a loopback interface. The advantage of this configuration is that the loopback interface is not affected by the effects of a flapping interface.
Step 8	Repeat Steps 6 and 7 to configure other BGP neighbors.	—

	Command or Action	Purpose
Step 9	**address-family l2vpn** [**vpls**] **Example:** `Device(config-router)# address-family l2vpn vpls`	Specifies the L2VPN address family and enters address family configuration mode. • The optional **vpls** keyword specifies that the VPLS endpoint provisioning information is to be distributed to BGP peers. • In this example, an L2VPN VPLS address family session is created.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 10.10.10.1 activate`	Enables the exchange of information with a BGP neighbor.
Step 11	**neighbor** {*ip-address* | *peer-group-name*} **send-community** {**both** | **standard** | **extended**} **Example:** `Device(config-router-af)# neighbor 10.10.10.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • In this example, an extended communities attribute is sent to the neighbor at 10.10.10.1.
Step 12	Repeat Steps 10 and 11 to activate other BGP neighbors under an L2VPN address family.	—
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.
Step 15	**show l2vpn vfi** **Example:** `Device# show l2vpn vfi`	Displays information about the Layer 2 VPN (L2VPN) virtual forwarding instances (VFI).
Step 16	**show ip bgp l2vpn vpls** {**all** | **rd** *route-distinguisher*} **Example:**	Displays information about the L2VPN VPLS address family.

	Command or Action	Purpose
	`Device# show ip bgp l2vpn vpls all`	

Customizing the VPLS Autodiscovery Settings

Several commands allow you to customize the Virtual Private LAN Service (VPLS) environment. You can specify identifiers for the VPLS domain, the route distinguisher (RD), the route target (RT), and the provider edge (PE) device. Perform this task to customize these identifiers.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **autodiscovery** **Example:** `Device(config)# l2 vfi vpls1 autodiscovery`	Enables VPLS Autodiscovery on the PE device and enters Layer 2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**vpls-id** {*autonomous-system-number*:*nn* | *ip-address*:*nn*} **Example:** `Device(config-vfi)# vpls-id 5:300`	(Optional) Assigns an identifier to the VPLS domain. • This command is optional because VPLS Autodiscovery automatically generates a VPLS ID using the Border Gateway Protocol (BGP) autonomous system (AS) number and the configured VFI VPN ID. You can use this command to change the automatically generated VPLS ID. • There are two formats for configuring the VPLS ID argument. It can be configured in the *autonomous-system-number:network number* (*ASN:nn*) format, as shown in the example, or it can be configured in the *IP-address:network number* format (*IP-address:nn*).

	Command or Action	Purpose
Step 6	**rd** {*autonomous-system-number*:*nn* | *ip-address*:*nn*} **Example:** `Device(config-vfi)# rd 2:3`	(Optional) Specifies the RD to distribute endpoint information. • This command is optional because VPLS Autodiscovery automatically generates an RD using the BGP autonomous system number and the configured VFI VPN ID. You can use this command to change the automatically generated RD. • There are two formats for configuring the route distinguisher argument. It can be configured in the *autonomous-system-number:network number* (*ASN:nn*) format, as shown in the example, or it can be configured in the *IP-address:network number* format (*IP-address:nn*).
Step 7	**route-target** [**import** | **export** | **both**] {*autonomous-system-number*:*nn* | *ip-address*:*nn*} **Example:** `Device(config-vfi)# route-target 600:2222`	(Optional) Specifies the RT. • This command is optional because VPLS Autodiscovery automatically generates an RT using the lower 6 bytes of the RD and the VPLS ID. You can use this command to change the automatically generated RT. • There are two formats for configuring the route target argument. It can be configured in the *autonomous-system-number:network number* (*ASN:nn*) format, as shown in the example, or it can be configured in the *IP-address:network number* format (*IP-address:nn*).
Step 8	**auto-route-target** **Example:** `Device(config-vfi)# auto-route-target`	(Optional) Enables the automatic generation of a RT.
Step 9	**end** **Example:** `Device(config-vfi)# end`	Exits L2 VFI configuration mode and returns to privileged EXEC mode. • Commands take effect after the device exits Layer 2 VFI configuration mode.

Customizing the VPLS Autodiscovery Settings using the commands associated with the L2VPN Protocol-Based CLIs feature

Several commands allow you to customize the Virtual Private LAN Service (VPLS) environment. You can specify identifiers for the VPLS domain, the route distinguisher (RD), the route target (RT), and the provider edge (PE) device. Perform this task to customize these identifiers.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes a L2VPN VFI context and enters L2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**autodiscovery bgp signaling** {**ldp** | **bgp**} **Example:** `Device(config-vfi)# autodiscovery bgp signaling ldp`	Enables the VPLS Autodiscovery: BGP Based feature on the PE device.
Step 6	**vpls-id** {*autonomous-system-number*:*nn* | *ip-address*:*nn*} **Example:** `Device(config-vfi)# vpls-id 5:300`	(Optional) Assigns an identifier to the VPLS domain. • This command is optional because VPLS Autodiscovery automatically generates a VPLS ID using the Border Gateway Protocol (BGP) autonomous system (AS) number and the configured VFI VPN ID. You can use this command to change the automatically generated VPLS ID. • There are two formats for configuring the VPLS ID argument. It can be configured in the

	Command or Action	Purpose
		autonomous-system-number:network number (*ASN:nn*) format, as shown in the example, or it can be configured in the *IP-address:network number* format (*IP-address:nn*).
Step 7	**rd** {*autonomous-system-number*:*nn* | *ip-address*:*nn*} **Example:** `Device(config-vfi)# rd 2:3`	(Optional) Specifies the RD to distribute endpoint information. • This command is optional because VPLS Autodiscovery automatically generates an RD using the BGP autonomous system number and the configured VFI VPN ID. You can use this command to change the automatically generated RD. • There are two formats for configuring the route distinguisher argument. It can be configured in the *autonomous-system-number:network number* (*ASN:nn*) format, as shown in the example, or it can be configured in the *IP-address:network number* format (*IP-address:nn*).
Step 8	**route-target** [**import** | **export** | **both**] {*autonomous-system-number*:*nn* | *ip-address*:*nn*} **Example:** `Device(config-vfi)# route-target 600:2222`	(Optional) Specifies the RT. • This command is optional because VPLS Autodiscovery automatically generates an RT using the lower 6 bytes of the RD and the VPLS ID. You can use this command to change the automatically generated RT. • There are two formats for configuring the route target argument. It can be configured in the *autonomous-system-number:network number* (*ASN:nn*) format, as shown in the example, or it can be configured in the *IP-address:network number* format (*IP-address:nn*).
Step 9	**auto-route-target** **Example:** `Device(config-vfi)# auto-route-target`	(Optional) Enables the automatic generation of a RT.
Step 10	**end** **Example:**	Exits L2 VFI configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-vfi)# end`	• Commands take effect after the device exits Layer 2 VFI configuration mode.

Configuring MST on VPLS N-PE Devices

A network-facing PE (N-PE) device is the root bridge for a Multiple Spanning Tree (MST) instance.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **manual** **Example:** `Device(config)# l2 vfi vpls-mst manual`	Creates a Layer 2 virtual forwarding instance (VFI) and enters Layer 2 VFI manual configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 4000`	Sets or updates the VPN ID on a VPN routing and forwarding (VRF) instance.
Step 5	**forward permit l2protocol all** **Example:** `Device(config-vfi)# forward permit l2protocol all`	Defines the VPLS pseudowire that is used to transport the bridge protocol data unit (BPDU) information between two N-PE devices.
Step 6	**neighbor** *peer-N-PE-ip-address* **encapsulation mpls** **Example:** `Device(config-vfi)# neighbor 10.76.100.12 encapsulation mpls`	Specifies the type of tunnel signaling and encapsulation mechanism for each VPLS peer.
Step 7	**exit** **Example:** `Device(config-vfi)# exit`	Exits Layer 2 VFI manual configuration mode and returns to global configuration mode.
Step 8	**spanning-tree mode** [**mst** | **pvst** | **rapid-pvst**] **Example:** `Device(config)# spanning-tree mode mst`	Switches between MST, Per-VLAN Spanning Tree+ (PVST+), and Rapid-PVST+ modes.

	Command or Action	Purpose
Step 9	**spanning-tree mst configuration** **Example:** `Device(config)# spanning-tree mst configuration`	Enters MST configuration mode.
Step 10	**name** *name* **Example:** `Device(config-mst)# name cisco`	Sets the name for the MST region.
Step 11	**revision** *version* **Example:** `Device(config-mst)# revision 11`	Sets the revision number for the MST configuration.
Step 12	**instance** *instance-id* **vlan** *vlan-range* **Example:** `Device(config-mst)# instance 1 vlan 100`	Maps a VLAN or a group of VLANs to an MST instance.
Step 13	**end** **Example:** `Device(config-mst)# end`	Exits MST configuration mode and enters privileged EXEC mode.
Step 14	**show spanning-tree mst** [*instance-id* [**detail**] [*interface*] | **configuration** [**digest**] | **detail** | **interface** *type number* [**detail**]] **Example:** `Device# show spanning-tree mst 1`	Displays information about the MST configuration.

Configuring MST on VPLS N-PE Devices using the commands associated with the L2VPN Protocol-Based CLIs feature

A network-facing PE (N-PE) device is the root bridge for a Multiple Spanning Tree (MST) instance.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls-mst`	Establishes an L2VPN VFI context and enters L2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 4000`	Sets or updates the VPN ID on a VPN routing and forwarding (VRF) instance.
Step 5	**forward permit l2protocol all** **Example:** `Device(config-vfi)# forward permit l2protocol all`	Defines the VPLS pseudowire that is used to transport the bridge protocol data unit (BPDU) information between two N-PE devices.
Step 6	**neighbor** *peer-N-PE-ip-address* **encapsulation mpls** **Example:** `Device(config-vfi)# neighbor 10.76.100.12 encapsulation mpls`	Specifies the type of tunnel signaling and encapsulation mechanism for each VPLS peer.
Step 7	**exit** **Example:** `Device(config-vfi)# exit`	Exits Layer 2 VFI manual configuration mode and returns to global configuration mode.
Step 8	**spanning-tree mode** [**mst** | **pvst** | **rapid-pvst**] **Example:** `Device(config)# spanning-tree mode mst`	Switches between MST, Per-VLAN Spanning Tree+ (PVST+), and Rapid-PVST+ modes.
Step 9	**spanning-tree mst configuration** **Example:** `Device(config)# spanning-tree mst configuration`	Enters MST configuration mode.
Step 10	**name** *name* **Example:** `Device(config-mst)# name cisco`	Sets the name for the MST region.
Step 11	**revision** *version* **Example:**	Sets the revision number for the MST configuration.

	Command or Action	Purpose
	`Device(config-mst)# revision 11`	
Step 12	**instance** *instance-id* **vlan** *vlan-range* **Example:** `Device(config-mst)# instance 1 vlan 100`	Maps a VLAN or a group of VLANs to an MST instance.
Step 13	**end** **Example:** `Device(config-mst)# end`	Exits MST configuration mode and enters privileged EXEC mode.
Step 14	**show spanning-tree mst** [*instance-id* [**detail**] [*interface*] | **configuration** [**digest**] | **detail** | **interface** *type number* [**detail**]] **Example:** `Device# show spanning-tree mst 1`	Displays information about the MST configuration.

Configuration Examples for VPLS Autodiscovery BGP Based

The following examples show the configuration of a network that uses VPLS Autodiscovery:

Example: Enabling VPLS Autodiscovery BGP Based

```
Device> enable
Device# configure terminal
Device(config)# l2 vfi vpls1 autodiscovery
Device(config-vfi)# vpn id 10
Device(config-vfi)# exit
```

Example: Enabling VPLS Autodiscovery BGP Based Using Commands Associated with L2VPN Protocol-Based Feature

Note For more information, see Configuring 802.1Q Access Ports for Tagged Traffic from a CE Device: Alternate Configuration.

```
Device> enable
Device# configure terminal
Device(config)# l2vpn vfi context vpls1
Device(config-vfi)# vpn id 10
```

```
Device(config-vfi)# autodiscovery bgp signaling ldp
Device(config-vfi)# exit
```

Example: Configuring BGP to Enable VPLS Autodiscovery

PE1

```
l2 router-id 10.1.1.1
l2 vfi auto autodiscovery
 vpn id 100
!
pseudowire-class mpls
 encapsulation mpls
!
interface Loopback1
 ip address 10.1.1.1 255.255.255.255
!
interface GigabitEthernet 0/0/1
 description Backbone interface
 ip address 192.168.0.1 255.255.255.0
 mpls ip
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
router bgp 1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 bgp update-delay 1
 neighbor 10.1.1.2 remote-as 1
 neighbor 10.1.1.2 update-source Loopback1
 neighbor 10.1.1.3 remote-as 1
 neighbor 10.1.1.3 update-source Loopback1
!
 address-family ipv4
 no synchronization
 no auto-summary
 exit-address-family
 !
 address-family l2vpn vpls
 neighbor 10.1.1.2 activate
 neighbor 10.1.1.2 send-community extended
 neighbor 10.1.1.3 activate
 neighbor 10.1.1.3 send-community extended
 exit-address-family
```

PE2

```
l2 router-id 10.1.1.2
l2 vfi auto autodiscovery
 vpn id 100
!
 pseudowire-class mpls
 encapsulation mpls
!
interface Loopback1
 ip address 10.1.1.2 255.255.255.255
!
interface GigabitEthernet 0/0/1
```

```
 description Backbone interface
 ip address 192.168.0.2 255.255.255.0
 mpls ip
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
router bgp 1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 bgp update-delay 1
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Loopback1
 neighbor 10.1.1.3 remote-as 1
 neighbor 10.1.1.3 update-source Loopback1
!
 address-family ipv4
 no synchronization
 no auto-summary
 exit-address-family
 !
 address-family l2vpn vpls
 neighbor 10.1.1.1 activate
 neighbor 10.1.1.1 send-community extended
 neighbor 10.1.1.3 activate
 neighbor 10.1.1.3 send-community extended
 exit-address-family
```

PE3

```
l2 router-id 10.1.1.3
l2 vfi auto autodiscovery
 vpn id 100
!
pseudowire-class mpls
 encapsulation mpls
!
interface Loopback1
 ip address 10.1.1.3 255.255.255.255
!
interface GigabitEthernet 0/0/1
 description Backbone interface
 ip address 192.168.0.3 255.255.255.0
 mpls ip
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
router bgp 1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 bgp update-delay 1
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Loopback1
 neighbor 10.1.1.2 remote-as 1
 neighbor 10.1.1.2 update-source Loopback1
!
 address-family ipv4
 no synchronization
 no auto-summary
```

```
 exit-address-family
 !
 address-family l2vpn vpls
 neighbor 10.1.1.1 activate
 neighbor 10.1.1.1 send-community extended
 neighbor 10.1.1.2 activate
 neighbor 10.1.1.2 send-community extended
 exit-address-family
```

Example: Configuring BGP to Enable VPLS Autodiscovery Using Commands Associated with L2VPN Protocol-Based Feature

Note For VPLS Autodiscovery with BGP signalling, see VPLS BGP Signaling.

PE1

```
l2vpn
 router-id 10.1.1.1
l2vpn vfi context auto
 vpn id 100
 autodiscovery bgp signaling ldp
!
interface pseudowire 1
 encapsulation mpls
 neighbor 33.33.33.33 1
!
interface Loopback1
 ip address 10.1.1.1 255.255.255.255
!
interface GigabitEthernet 0/0/1
 description Backbone interface
 ip address 192.168.0.1 255.255.255.0
 mpls ip
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
router bgp 1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 bgp update-delay 1
 neighbor 10.1.1.2 remote-as 1
 neighbor 10.1.1.2 update-source Loopback1
 neighbor 10.1.1.3 remote-as 1
 neighbor 10.1.1.3 update-source Loopback1
!
 address-family ipv4
 no synchronization
 no auto-summary
 exit-address-family
 !
 address-family l2vpn vpls
 neighbor 10.1.1.2 activate
 neighbor 10.1.1.2 send-community extended
 neighbor 10.1.1.3 activate
```

```
 neighbor 10.1.1.3 send-community extended
 exit-address-family
```

PE2

```
l2vpn
 router-id 10.1.1.2
l2vpn vfi context auto
 vpn id 100
 autodiscovery bgp signaling ldp

!
 interface pseudowire 1
 encapsulation mpls
 neighbor 33.33.33.33 1
!
interface Loopback1
 ip address 10.1.1.2 255.255.255.255
!
interface GigabitEthernet 0/0/1
 description Backbone interface
 ip address 192.168.0.2 255.255.255.0
 mpls ip
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
router bgp 1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 bgp update-delay 1
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Loopback1
 neighbor 10.1.1.3 remote-as 1
 neighbor 10.1.1.3 update-source Loopback1
!
 address-family ipv4
 no synchronization
 no auto-summary
 exit-address-family
 !
 address-family l2vpn vpls
 neighbor 10.1.1.1 activate
 neighbor 10.1.1.1 send-community extended
 neighbor 10.1.1.3 activate
 neighbor 10.1.1.3 send-community extended
 exit-address-family
```

PE3

```
l2vpn
 router-id 10.1.1.3
l2vpn vfi context auto
 vpn id 100
 autodiscovery bgp signaling ldp

!
interface pseudowire 1
 encapsulation mpls
 neighbor 33.33.33.33 1
!
```

```
interface Loopback1
 ip address 10.1.1.3 255.255.255.255
!
interface GigabitEthernet 0/0/1
 description Backbone interface
 ip address 192.168.0.3 255.255.255.0
 mpls ip
!
router ospf 1
 log-adjacency-changes
 network 10.1.1.0 0.0.0.255 area 0
 network 172.16.0.0 0.0.0.255 area 0
!
router bgp 1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 bgp update-delay 1
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Loopback1
 neighbor 10.1.1.2 remote-as 1
 neighbor 10.1.1.2 update-source Loopback1
!
 address-family ipv4
 no synchronization
 no auto-summary
 exit-address-family
 !
 address-family l2vpn vpls
 neighbor 10.1.1.1 activate
 neighbor 10.1.1.1 send-community extended
 neighbor 10.1.1.2 activate
 neighbor 10.1.1.2 send-community extended
 exit-address-family
```

Example: Customizing VPLS Autodiscovery Settings

```
Device> enable
Device# configure terminal
Device(config)# l2 vfi vpls1 autodiscovery
Device(config-vfi)# vpn id 10
Device(config-vfi)# vpls-id 5:300
Device(config-vfi)# rd 2:3
Device(config-vfi)# route-target 600:2222
Device(config-vfi)# end
```

Example: Customizing VPLS Autodiscovery Settings using the commands associated with the L2VPN Protocol-Based CLIs feature

```
Device> enable
Device# configure terminal
Device(config)# l2vpn vfi context vpls1
Device(config-vfi)# vpn id 10
Device(config-vfi)# autodiscovery bgp signaling ldp
Device(config-vfi)# vpls-id 5:300
Device(config-vfi)# rd 2:3
Device(config-vfi)# route-target 600:2222
Device(config-vfi)# end
```

Example: Configuring MST on VPLS N-PE Devices

```
Device> enable
Device# configure terminal
Device(config)# l2 vfi vpls-mst manual
Device(config-vfi)# vpn id 4000
Device(config-vfi)# forward permit l2protocol all
Device(config-vfi)# neighbor 10.76.100.12 encapsulation mpls
Device(config-vfi)# exit
Device(config)# spanning-tree mode mst
Device(config)# spanning-tree mst configuration
Device(config-mst)# name cisco
Device(config-mst)# revision 11
Device(config-mst)# instance 1 vlan 100
Device(config-mst)# end
```

The following is sample output from the **show spanning-tree mst** command:

```
Device# show spanning-tree mst 1

##### MST1    vlans mapped:   100
Bridge        address 0023.3380.f8bb  priority      4097  (4096 sysid 1)
Root          this switch for MST1                              // Root for MST instance
 1 with VLAN 100
Interface                     Role Sts Cost      Prio.Nbr Type
-----------                ---- -- ------ ---- --------------------------------
Gi1/0/0                       Desg FWD 20000     128.18   P2p   // Access interface
VPLS-MST                      Desg FWD 1         128.28   Shr   // Forward VFI
```

The following is sample output from the **show spanning-tree mst detail** command:

```
Device# show spanning-tree mst 1 detail

##### MST1    vlans mapped:   100
Bridge        address 0023.3380.f8bb  priority      4097  (4096 sysid 1)
Root          this switch for MST1            // Root for MST instance 1 with VLAN 100
GigabitEthernet1/0/0 of MST1 is designated forwarding
Port info             port id         128.18  priority    128  cost       20000
Designated root       address 0023.3380.f8bb  priority   4097  cost           0
Designated bridge     address 0023.3380.f8bb  priority   4097  port id   128.18
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 40, received 5
VPLS-4000 of MST1 is designated forwarding
Port info             port id         128.28  priority    128  cost           1
Designated root       address 0023.3380.f8bb  priority   4097  cost           0
Designated bridge     address 0023.3380.f8bb  priority   4097  port id   128.28
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 28, received 26         // BPDU message exchange between N-PE devices
```

Example: Configuring MST on VPLS N-PE Devices using the commands associated with the L2VPN Protocol-Based CLIs feature

```
Device> enable
Device# configure terminal
Device(config)# l2vpn vfi context vpls-mst
Device(config-vfi)# vpn id 4000
Device(config-vfi)# forward permit l2protocol all
Device(config-vfi)# member 10.76.100.12 encapsulation mpls
Device(config-vfi)# exit
```

```
Device(config)# spanning-tree mode mst
Device(config)# spanning-tree mst configuration
Device(config-mst)# name cisco
Device(config-mst)# revision 11
Device(config-mst)# instance 1 vlan 100
Device(config-mst)# end
```

The following is sample output from the **show spanning-tree mst** command:

```
Device# show spanning-tree mst 1

##### MST1    vlans mapped:   100
Bridge        address 0023.3380.f8bb  priority      4097  (4096 sysid 1)
Root          this switch for MST1                            // Root for MST instance
 1 with VLAN 100
Interface                       Role Sts Cost      Prio.Nbr Type
-----------                  --- -- ------ ---- --------------------------------
Gi1/0/0                         Desg FWD 20000     128.18   P2p   // Access interface
VPLS-MST                        Desg FWD 1         128.28   Shr   // Forward VFI
```

The following is sample output from the **show spanning-tree mst detail** command:

```
Device# show spanning-tree mst 1 detail

##### MST1    vlans mapped:   100
Bridge        address 0023.3380.f8bb  priority      4097  (4096 sysid 1)
Root          this switch for MST1            // Root for MST instance 1 with VLAN 100
GigabitEthernet1/0/0 of MST1 is designated forwarding
Port info             port id         128.18  priority    128  cost       20000
Designated root       address 0023.3380.f8bb  priority   4097  cost           0
Designated bridge     address 0023.3380.f8bb  priority   4097  port id   128.18
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 40, received 5
VPLS-4000 of MST1 is designated forwarding
Port info             port id         128.28  priority    128  cost           1
Designated root       address 0023.3380.f8bb  priority   4097  cost           0
Designated bridge     address 0023.3380.f8bb  priority   4097  port id   128.28
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 28, received 26        // BPDU message exchange between N-PE devices
```

Example: BGP VPLS Autodiscovery Support on Route Reflector

In the following example, a host named PE-RR (indicating Provider Edge-Route Reflector) is configured as a route reflector that is capable of reflecting Virtual Private LAN Service (VPLS) prefixes. The VPLS address family is configured using the **address-family l2vpn vpls** command.

```
hostname PE-RR
!
router bgp 1
 bgp router-id 10.1.1.3
 no bgp default route-target filter
 bgp log-neighbor-changes
neighbor iBGP-PEERS peer-group
neighbor iBGP-PEERS remote-as 1
neighbor iBGP-PEERS update-source Loopback1
neighbor 10.1.1.1 peer-group iBGP-PEERS
neighbor 10.1.1.2 peer-group iBGP-PEERS
!
address-family l2vpn vpls
  neighbor iBGP-PEERS send-community extended
  neighbor iBGP-PEERS route-reflector-client
  neighbor 10.1.1.1 peer-group iBGP-PEERS
```

```
  neighbor 10.1.1.2 peer-group iBGP-PEERS
exit-address-family
```

Additional References for VPLS Autodiscovery BGP Based

Related Documents

Related Topic	Document Title
Cisco IOS commands	Master Command List, All Releases
MPLS commands	Multiprotocol Label Switching Command Reference

Standards and RFCs

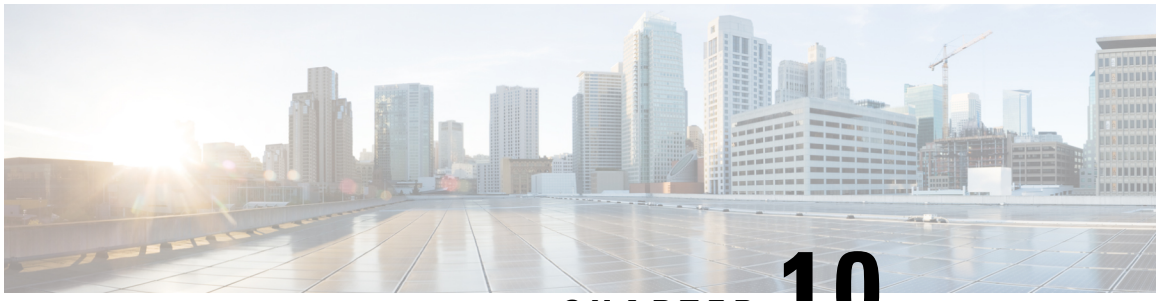
Standard/RFC	Title
draft-ietf-l2vpn-signaling-08.txt	*Provisioning, Autodiscovery, and Signaling in L2VPNs*
draft-ietf-l2vpn-vpls-bgp-08.8	*Virtual Private LAN Service (VPLS) Using BGP for Autodiscovery and Signaling*
draft-ietf-mpls-lsp-ping-03.txt	*Detecting MPLS Data Plane Failures*
draft-ietf-pwe3-vccv-01.txt	*Pseudo-Wire (PW) Virtual Circuit Connection Verification (VCCV)*
RFC 3916	*Requirements for Pseudo-wire Emulation Edge-to-Edge (PWE3)*
RFC 3981	*Pseudo Wire Emulation Edge-to-Edge Architecture*
RFC 6074	Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)
RFC 4761	Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling

MIBs

MIB	MIBs Link
• CISCO-IETF-PW-ATM-MIB (PW-ATM-MIB) • CISCO-IETF-PW-ENET-MIB (PW-ENET-MIB) • CISCO-IETF-PW-FR-MIB (PW-FR-MIB) • CISCO-IETF-PW-MIB (PW-MIB) • CISCO-IETF-PW-MPLS-MIB (PW-MPLS-MIB)	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register on Cisco.com.	http://www.cisco.com/techsupport



CHAPTER 10

VPLS BGP Signaling

The two primary functions of the Virtual Private LAN Service (VPLS) control plane are autodiscovery and signaling. The VPLS BGP Signaling feature enables you to use BGP as both an autodiscovery and a signaling protocol for VPLS, in accordance with RFC 4761.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for VPLS BGP Signaling

You are familiar with the concepts in the “Configuring Virtual Private LAN Services” and the “VPLS Autodiscovery BGP Based” modules of the *MPLS Layer 2 VPNs Configuration Guide* .

Information About VPLS BGP Signaling

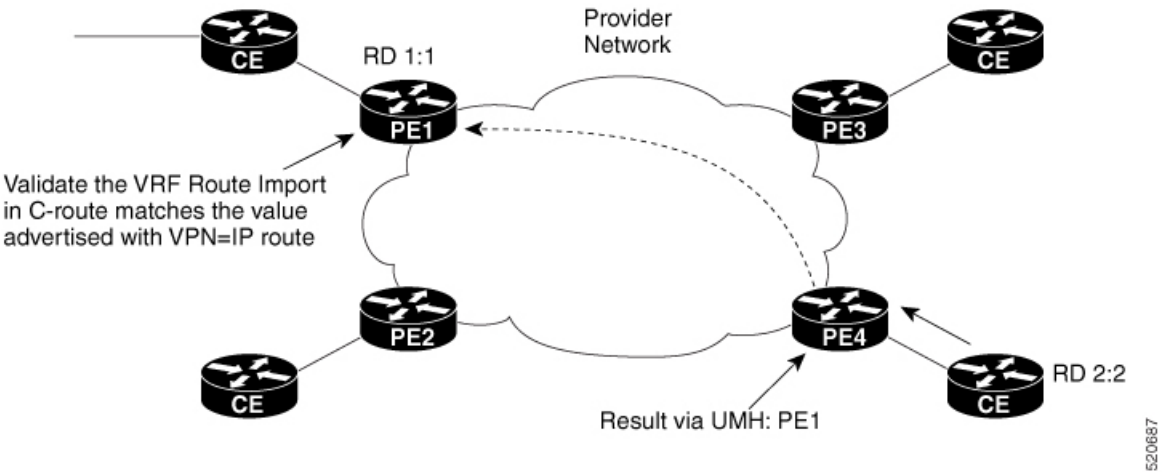
Overview of VPLS BGP Signaling

Prior to the VPLS BGP Signaling feature, BGP was used for autodiscovery and Label Distribution Protocol (LDP) for signaling in accordance with RFC 6074. The VPLS BGP Signaling feature enables you to use BGP as the control plane protocol for both autodiscovery and signaling in accordance with RFC 4761.

As specified in RFC 4761, internal BGP (iBGP) peers will exchange update messages of the L2VPN AFI/SAFI with L2VPN information to perform both autodiscovery and signaling. The BGP multiprotocol Network Layer Reachability Information (NLRI) consists of a Route Distinguisher (RD), VPLS Endpoint ID (VE ID), VE Block Offset (VBO), VE Block Size (VBS), and Label Base (LB).

The figure below shows the format of the NLRI for RFC 4761.

Figure 13: RFC 4761 NLRI



Additional information, such as next-hop, route target (specified for a VPLS instance), and other Layer 2 data are carried in the BGP extended community attributes. A route target-based import/export mechanism similar to L3VPN is performed by BGP to filter L2VPN NLRIs of a particular VPLS instance.

Whether you use BGP signaling (RFC 4761) or LDP signaling (RFC 6074) depends on the commands you specify. To enable the VPLS BGP Signaling feature, use the **autodiscovery bgp signaling bgp** command in L2 VFI configuration mode. This command is supported on a per VPLS instance basis.

If a BGP session receives an invalid (that is, not matching the configuration) BGP update advertisement (update or withdraw), it is ignored.

BGP's main task in supporting VPLS is route distribution via the L2VPN address family and interactions with L2VPN. Interactions between BGP and other components remain the same. Basic BGP functionalities like best-path selection, next-hop handling, and update generation, continue to operate in the same manner with VPLS BGP signaling. BGP RT constraint works seamlessly with the BGP VPLS Signaling feature.

The above example shows sample configuration on one PE. Similar configuration can be mirrored on other PEs.

How to Configure VPLS BGP Signaling

Configuring VPLS BGP Signaling

Before you begin

Note For more information, see *Configuring Virtual Private Lan Services*.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **l2vpn vfi context** *name*

Example:

```
Device(config)# l2vpn vfi context vfi1
```

Establishes a L2VPN virtual forwarding interface (VFI) between two or more separate networks and enters Layer 2 VFI configuration mode.

Step 4 **vpn id** *vpn-id*

Example:

```
Device(config-vfi)# vpn id 100
```

Configures a VPN ID for the VPLS domain.

Step 5 **autodiscovery bgp signaling** {**bgp** | **ldp**} [**template** *template-name*]

Example:

```
Device(config-vfi)# autodiscovery bgp signaling bgp
```

Enables BGP signaling and discovery or LDP signaling and enters L2VPN VFI autodiscovery configuration mode.

Note For the VPLS BGP Signaling feature use the **autodiscovery bgp signaling bgp** command.

Step 6 **ve id** *ve-id*

Example:

```
Device(config-vfi-autodiscovery)# ve id 1001
```

Specifies the VPLS endpoint (VE) device ID value. The VE ID identifies a VFI within a VPLS service. The VE device ID value is from 1 to 16384.

Step 7 **ve range** *ve-range*

Example:

```
Device(config-vfi-autodiscovery)# ve range 12
```

Specifies the VE device ID range value. The VE range overrides the minimum size of VE blocks. The default minimum size is 10. Any configured VE range must be higher than 10.

Step 8 **exit**

Example:

```
Device(config-vfi-autodiscovery)# exit
```

Exits L2VPN VFI autodiscovery configuration mode and enters L2VPN VFI configuration mode.

Step 9 **exit**

Example:

```
Device(config-vfi)# exit
```

Exits L2VPN VFI configuration mode and enters global configuration mode.

Step 10 **router bgp** *autonomous-system-number*

Example:

```
Device(config)# router bgp 100
```

Enters router configuration mode to create or configure a BGP routing process.

Step 11 **bgp graceful-restart**

Example:

```
Device(config-router)# bgp graceful-restart
```

Enables the BGP graceful restart capability and BGP nonstop forwarding (NSF) awareness.

Step 12 **neighbor** *ip-address* **remote-as** *autonomous-system-number*

Example:

```
Device(config-router)# neighbor 10.10.10.1 remote-as 100
```

Configures peering with a BGP neighbor in the specified autonomous system.

Step 13 **address-family l2vpn** [**vpls**]

Example:

```
Device(config-router)# address-family l2vpn vpls
```

Specifies the L2VPN address family and enters address family configuration mode.

- The optional **vpls** keyword specifies that VPLS endpoint provisioning information is to be distributed to BGP peers.

In this example, an L2VPN VPLS address family session is created.

Step 14 **neighbor** *ip-address* **activate**

Example:

```
Device(config-router-af)# neighbor 10.10.10.1 activate
```

Enables the neighbor to exchange information for the L2VPN VPLS address family with the local device.

Step 15 **neighbor** *ip-address* **send-community** [**both** | **standard** | **extended**]

Example:

```
Device(config-router-af)# neighbor 10.10.10.1 send-community extended
```

Specifies that a communities attribute should be sent to a BGP neighbor.

- In this example, an extended communities attribute is sent to the neighbor at 10.10.10.1.

Step 16 **neighbor** *ip-address* **suppress-signaling-protocol ldp**

Example:

```
Device(config-router-af)# neighbor 10.10.10.1 suppress-signaling-protocol ldp
```

Suppresses LDP signaling and enables BGP signaling.

- In this example LDP signaling is suppressed (and BGP signaling enabled) for the neighbor at 10.10.10.1.

Step 17 **end**

Example:

```
Device(config-router-af)# end
```

Exits address family configuration mode and returns to privileged EXEC mode.

Step 18 **show bgp l2vpn vpls** {**all** | **rd** *route-distinguisher*}

Example:

```
Device# show bgp l2vpn vpls all
```

(Optional) Displays information about the L2VPN VPLS address family.

Configuration Examples for VPLS BGP Signaling

Example: Configuring and Verifying VPLS BGP Signaling

```
l2vpn vfi context vfi1
 vpn id 100
 autodiscovery bgp signaling bgp
  ve id 1001
  ve range 10
 !
!
router bgp 100
 bgp graceful-restart
 neighbor 192.168.200.224 remote-as 100
 neighbor 192.168.200.224 update-source Loopback1
 !
 address-family l2vpn vpls
  neighbor 192.168.200.224 activate
  neighbor 192.168.200.224 send-community extended
  neighbor 192.168.200.224 suppress-signaling-protocol ldp
  exit-address-family
 !
show bgp l2vpn vpls all

Network                          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:100
*>100:100:VEID-1001:Blk-1001/136    10.0.0.0                             32768  ?

*>i 100:100:VEID-1003:Blk-1000/136  192.168.200.224              0      100     0     ?
```

Additional References for VPLS BGP Signaling

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
BGP commands: complete command syntax, command mode, defaults, command history, usage guidelines, and examples.	Cisco IOS IP Routing: BGP Command Reference
Configuring Virtual Private LAN Services	*MPLS Layer 2 VPNs Configuration Guide* MPLS Layer 2 VPNs Configuration Guide

Related Topic	Document Title
Configuring Access Port	Configuring Virtual Private LAN Services, *MPLS Layer 2 VPNs Configuration Guide* MPLS Layer 2 VPNs Configuration Guide
VPLS Autodiscovery BGP Based	*MPLS Layer 2 VPNs Configuration Guide* MPLS Layer 2 VPNs Configuration Guide

Standards and RFCs

Standard/RFC	Title
RFC 4761	*Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*
RFC 6074	*Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

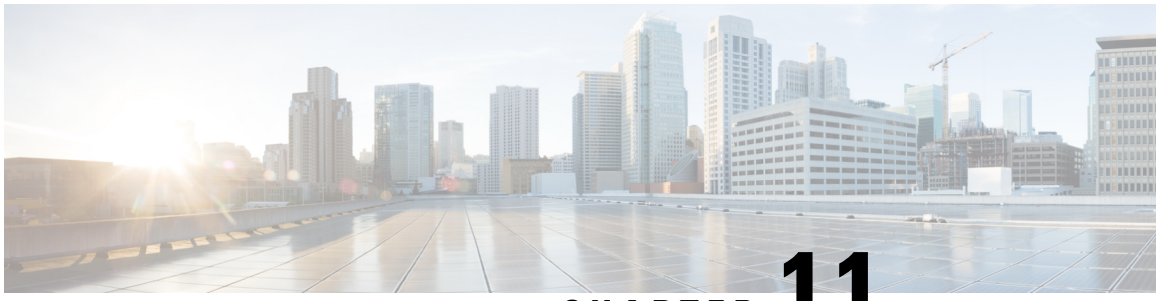
Feature Information for VPLS BGP Signaling

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 9: Feature Information for VPLS BGP Signaling

Feature Name	Releases	Feature Information
VPLS BGP Signaling		The VPLS BGP Signaling feature enables you to use BGP as both an autodiscovery and signaling protocol for VPLS, in accordance with RFC 4761. The following commands were introduced or modified: **autodiscovery (MPLS)**, **neighbor suppress-signaling-protocol**, **show bgp l2vpn vpls**, and **ve**. In Cisco IOS XE Release 3.8S, support was added for the Cisco ASR 903 router.



CHAPTER 11

N:1 PVC Mapping to PWE with Nonunique VPIs

The N:1 PVC Mapping to PseudoWire Emulation (PWE) with Nonunique virtual path identifiers (VPIs) feature maps one or more ATM permanent virtual circuits (PVCs) to a single pseudowire (PW). There are two modes of AAL0 encapsulation, N:1 and 1:1 mapping. In N:1 mapping, multiple unrelated virtual path identifier/virtual channel identifier (VPI/VCI) are carried over a single Multiprotocol Label Switching (MPLS) PW. This is an efficient mapping method because less resources are used from the MPLS network. In 1:1 mapping, a single VPI/VCI is carried over a single MPLS PW. Benefits of this feature include the following:

- Aggregate quality of service (QoS) can be applied to related PVCs.
- Bandwidth is conserved with the reduction in the number of pseudowires that are used.

Restrictions for N:1 PVC Mapping to PWE with Nonunique VPIs

- N:1 permanent virtual circuits (PVC) mapping configuration is supported only on multipoint subinterfaces; it is not supported on main interfaces or point-to-point subinterfaces.
- N:1 PVC mapping mode is not supported on Access Circuit Redundancy subinterfaces.
- Preconfigured PVCs cannot exist on the multipoint subinterface on which you want to configure N:1 PVC mapping.
- An attachment circuit that has been bound to a pseudowire cannot be removed unless all Layer 2 virtual circuits (VCs) have been removed.
- Layer 3 PVCs cannot be configured on N:1 subinterfaces.
- Cell packing values configured under a VC class attached to the PVC, main interface, or subinterface will not be inherited by N:1 PVCs.
- Operation, Administration, and Maintenance (OAM) functionality is not supported on N:1 Layer 2 PVCs. OAM cells coming from the customer edge (CE) network will be treated as normal data traffic and will traverse through the pseudowire.

- Only ATM adaptation layer type 0 (AAL0) encapsulation is supported for N:1 PVCs.
- The service policy configuration can be configured only at the subinterface level for N:1 PVCs.
- ATM N:1 and PVP modes cannot be configured on different subinterfaces that belong to a physical interface.
- You cannot change the ATM interface mode from point-to-point to multipoint or from multipoint to point-to-point.
- If you change a layer 2 ATM interface to a layer 3 ATM interface, traffic will not flow.

Information About N:1 PVC Mapping to PWE with Nonunique VPIs

N:1 PVC Mapping to PWE with Nonunique VPIs Feature Description

To transport ATM cells over Multiprotocol Label Switching (MPLS), a VC is established between the provider edge (PE) routers on both ends of the MPLS backbone. With the N:1 permanent virtual circuit (PVC) Mapping to PseudoWire Emulation (PWE) with Nonunique VPIs feature, multiple PVCs irrespective of their Virtual Path Identifiers (VPIs), are transported over a single pseudowire configured on a subinterface. ("N:1" refers to the number of PVCs transported over one pseudowire). ATM cells are packed together in a single frame and sent over the single pseudowire. The ATM cell header information is packed together with the cell payload on a per-cell basis in the packets so that packets received at the egress end are unpacked and the ATM cells are mapped to the respective PVCs.

In N:1 PVC mapping mode, the device can pack cells only from a single PVC in an MPLS packet to transmit over a pseudowire; cells from multiple PVCs cannot be packed in a single MPLS packet and mapped to a single pseudowire for transmission. However, if a device receives an MPLS packet that is packed with cells from multiple PVCs, then those cells will be unpacked and sent to the respective PVCs.

How to Configure N:1 PVC Mapping to PWE with Nonunique VPIs

Configuring N:1 PVC Mapping to PWE with Nonunique VPIs

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface atm** *slot/subslot/port* **Example:** `Device(config)# interface atm 9/1/1`	Enables the ATM interface and enters interface configuration mode.
Step 4	**atm mcpt-timers** *timer1 timer2 timer3* **Example:** `Device(config-if)# atm mcpt-timers 100 200 300`	Sets the Maximum Cell Packing Timeout (MCPT) values in microseconds. • The MCPT timer sets the time for which the device waits for the raw cells (AAL0 encapsulation) to be packed into a single packet for punting to the pseudowire.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 6	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 7	**interface atm** *slot/subslot/port.subslot* **multipoint** **Example:** `Device(config)# interface atm 9/1/1.1 multipoint`	Enters subinterface configuration mode and creates a multipoint subinterface on the given port on the specified ATM Shared Port Adapter (SPA).
Step 8	**no ip address** **Example:** `Device(config-subif)# no ip address`	Removes the interface IP address.
Step 9	**atm enable-ilmi-trap** **Example:** `Device(config-subif)# atm enable-ilmi-trap`	Generates an Integrated Local Management Interface (ILMI) atmfVccChange trap when an ATM interface or subinterface is enabled or shut down.
Step 10	**cell-packing** *maxcells* **mcpt-timer** *timer-number* **Example:** `Device(config-subif)# cell-packing 20 mcpt-timer 2`	Enables ATM over MPLS to pack multiple ATM cells into each MPLS packet within the MCPT timing.

	Command or Action	Purpose
Step 11	**xconnect** *peer-ipaddress vc-id* **encapsulation mpls** **Example:** `Device(config-subif)# xconnect 10.1.1.1 100 encapsulation mpls`	(Optional) Enables the attachment circuit and specifies the IP address of the peer, a VC ID, and the data encapsulation method.
Step 12	**pvc** *vpi/vci* **l2transport** **Example:** `Device(config-subif)# pvc 10/100 l2transport`	Assigns a VPI and virtual channel identifier (VCI).
Step 13	Repeat Step 12 for the number of PVCs that you want to configure.	—
Step 14	**end** **Example:** `Device(config-subif)# end`	Exits subinterface configuration mode and returns to privileged EXEC mode.

Configuration Examples for N:1 PVC Mapping to PWE with Nonunique VPIs

Example: Configuring N:1 PVC Mapping to PWE with Nonunique VPIs

The following example shows how to configure the N:1 ATM permanent virtual circuit (PVC) mapping to pseudowires with non unique virtual path identifiers (VPIs):

```
Device> enable
Device# configure terminal
Device(config)# interface atm 0/1/0
Device(config-if)# atm mcpt-timers 500 5000 50000
Device(config-if)# exit
Device# configure terminal
Device(config)# interface atm 0/1/0.1 multipoint
Device(config-subif)# no ip address
Device(config-subif)# atm enable-ilmi-trap
Device(config-subif)# cell packing 20 mcpt-timer 2
Device(config-subif)# xconnect 10.1.1.1 100 encapsulation mpls
Device(config-subif)# pvc 10/100 l2transport
Device(config-subif)# pvc 11/122 l2transport
Device(config-subif)# pvc 19/231 l2transport
Device(config-subif)# end
```

Verifying the N:1 PVC Mapping to PWE with Nonunique VPIs Configuration

To verify the N:1 PVC Mapping to PWE with Nonunique VPIs Configuration, use the **show mpls l2transport vc** command in user EXEC or privileged EXEC mode.

```
Router# show mpls l2transport vc

Local intf      Local circuit              Dest address    VC ID    Status
------------    -------------------------  --------------  -------  --------
AT0/1/1.1         ATM CELL ATM0/1/1.1         2.2.2.2       100      UP

interface ATM0/0/0.1/1/1/1
atm mcpt-timers 20 30 40

interface ATM0/0/0.1/1/1/1.1 multipoint
no ip address
no atm enable-ilmi-trap
cell-packing 2 mcpt-timer 1
xconnect 2.2.2.2 100 encapsulation mpls
pvc 10/100 l2transport
pvc 20/200 l2transport
pvc 30/300 l2transport
```

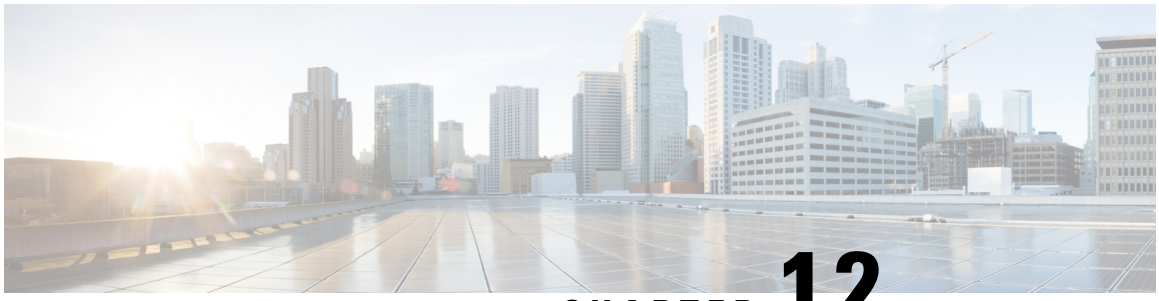
Additional References

Related Documents

Related Topic	Document Title
ATM commands	Asynchronous Transfer Mode Command Reference

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 12

Pseudowire Group Switchover

The Pseudowire Group Switchover feature allows all pseudowires in a group to be quickly switched over to backup pseudowires. This group switchover is triggered by a single “group down” status message received from a remote peer.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Prerequisites for Pseudowire Group Switchover

- The remote provider edge (PE) router must be capable of sending group status messages.
- Label Distribution Protocol (LDP) must be implemented on the network.
- Each xconnect must have a backup pseudowire configured.

Restrictions for Pseudowire Group Switchover

This feature is supported on the following attachment circuits:

- Ethernet VLAN
- Asynchronous Transfer Mode (ATM)
- Circuit Emulation (CEM) over MPLS
- The pseudowire group switch over convergence number increments linearly with thousand virtual circuits taking 16 seconds of convergence time.

Information About Pseudowire Group Switchover

Introduction to Pseudowire Group Switchover

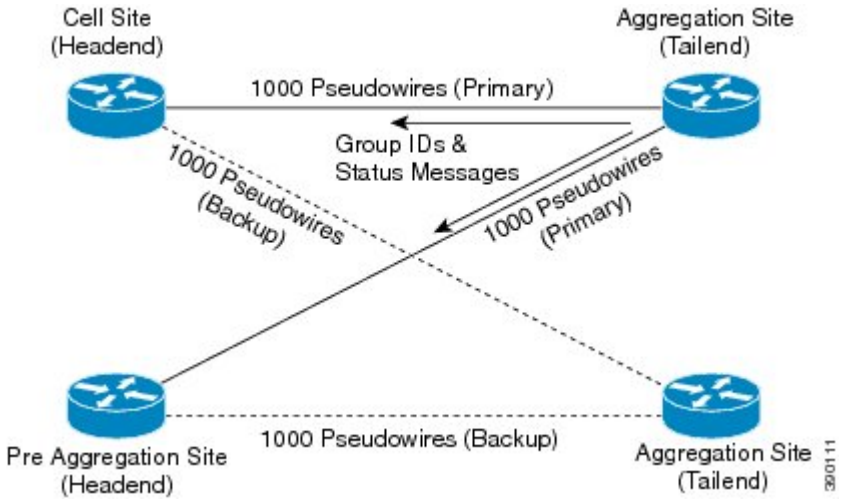
The Pseudowire Group Switchover feature allows you to reduce the switchover time from main pseudowires to backup pseudowires when a fault is encountered. The reduced switchover time is achieved by grouping Label Distribution Protocol (LDP) status messages and internal interprocess communication (IPC) messages.

When the remote peer detects an attachment circuit failure, it sends an LDP status message. When this status message is received, the designated backup pseudowires take over. Packets are then routed through the backup pseudowires.

Pseudowires can be grouped together by assigning a group ID. When an LDP status message is received by a pseudowire group, the entire group switches over, thus reducing switchover time.

Note The Pseudowire Group Switchover feature is enabled by default and cannot be disabled.

Figure 14: Primary and Backup Pseudowire Groups



How to Configure Predictive Switchover

Predictive switchover allows switchovers from a main pseudowire to a backup pseudowire with a remote "standby" status, without waiting for an “up” status from the remote peer.

Predictive switchover is configured by enabling redundancy predictive mode in global configuration mode or xconnect configuration mode.

Configuring Predictive Switchover (Global Configuration Mode)

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn** **Example:** `Device(config)# l2vpn`	Enters l2vpn configuration mode.
Step 4	**redundancy predictive enabled** **Example:** `Device(config-l2vpn)# redundancy predictive enabled`	Enables redundancy predictive mode. • By default, redundancy predictive mode is disabled.
Step 5	**end** **Example:** `Device(config-l2vpn)# end`	Exits l2vpn configuration mode and returns to privileged EXEC mode.

Configuring Predictive Switchover (Xconnect Configuration Mode)

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2vpn xconnect context** *context-name* **Example:** `Device(config)# l2vpn xconnect context con1`	Creates an L2VPN cross-connect context and enters xconnect configuration mode.
Step 4	**redundancy predictive enabled** **Example:** `Device(config-xconnect)# redundancy predictive enabled`	Enables redundancy predictive mode.
Step 5	**end** **Example:** `Device(config-xconnect)# end`	Exits xconnect configuration mode and returns to privileged EXEC mode.

Verifying a Pseudowire Group Switchover Configuration

You can use **show** commands to view information about a pseudowire group switchover configuration.

The following example shows how to display information about Any Transport over MPLS (AToM) virtual circuits (VCs):

```
Device# show l2vpn atom vc destination 2.1.1.2 group remote 6

                                      Service
Interface  Dest Address     VC ID     Type     Name                    Status
---------  ---------------  ---------- ------ ------------------------ ----------
pw100001   2.1.1.2         1234000    p2p      Et1/0.1-1001            UP
```

The following example shows how to display the status of the pseudowire switching point:

```
Device# show l2vpn atom vc destination 2.1.1.2 group remote 6 detail

pseudowire100001 is up, VC status is up PW type: Ethernet
  Create time: 5d20h, last status change time: 5d20h
    Last label FSM state change time: 5d20h
  Destination address: 2.1.1.2 VC ID: 1234000
    Output interface: Et0/0, imposed label stack {2001}
  Preferred path: not configured
  Default path: active
  Next hop: 20.0.0.2
Member of xconnect service Et1/0.1-1001, group right
  Associated member Et1/0.1 is up, status is up
  Interworking type is Ethernet
  Service id: 0x6d000002
Signaling protocol: LDP, peer 2.1.1.2:0 up
  Targeted Hello: 10.1.1.1(LDP Id) -> 2.1.1.2, LDP is UP
  Graceful restart: not configured and not enabled
  Non stop routing: not configured and not enabled
  PWid FEC (128), VC ID: 1234000
```

```
  Status TLV support (local/remote) : enabled/supported
    LDP route watch : enabled
    Label/status state machine : established, LruRru
    Local dataplane status received : No fault
    BFD dataplane status received : Not sent
    BFD peer monitor status received : No fault
    Status received from access circuit : No fault
    Status sent to access circuit : No fault
    Status received from pseudowire i/f : No fault
    Status sent to network peer : No fault
    Status received from network peer : No fault
    Adjacency status of remote peer : No fault
Sequencing: receive disabled, send disabled
Bindings
Parameter    Local                          Remote
------------ ------------------------------ ------------------------------
Label        2007                           2001
Group ID     0                              6
Interface
MTU          1500                           1500
Control word on (configured: autosense)     on
PW type      Ethernet                       Ethernet
VCCV CV type 0x12                           0x12
               LSPV [2], BFD/Raw [5]          LSPV [2], BFD/Raw [5]
VCCV CC type 0x07                           0x07
             CW [1], RA [2], TTL [3]        CW [1], RA [2], TTL [3]
Status TLV   enabled                        supported
Dataplane:
  SSM segment/switch IDs: 12309/4115 (used), PWID: 1
Rx Counters
  106563 input transit packets, 9803650 bytes
  0 drops, 0 seq err
Tx Counters
  0 output transit packets, 0 bytes
  0 drops
```

The following example lists the active and standby segment pairs associated with each peer IP address and group identifier:

```
Device# show ssm group

Active              Standby
IP Address       Group ID       Segment/Switch      Segment/Switch
=============================================================================
2.1.1.2              6            8215/4115           4116/8210
```

The following example displays the number of active and standby segment pairs associated with each peer IP address and group identifier:

```
Device# show ssm group 2.1.1.2 6 summary

IP Address        Group ID      Group Members
=============================================
2.1.1.2               6              1
```

The following example displays the number of pseudowires programmed in the hardware, with grouping information:

```
Device# show platform hardware pp active pw eompls group brief

Brief L2VPN EoMPLS Pseudo Wire Group Info

IP address           Group ID            Count
-----------------------------------------------------
0x47474747           100695488            90
```

Troubleshooting a Pseudowire Group Switchover Configuration

Use the **debug platform software atom brief** command to view information about the following configurations:

- Add Group
- Delete From Group
- Group Switchovers

Note We recommend that you use the **debug platform software atom brief** command only under Cisco Technical Assistance Center (TAC) supervision.

Configuration Examples for Predictive Switchover

Example: Configuring Predictive Switchover (Global Configuration Mode)

```
Device> enable
Device# configure terminal
Device(config)# l2vpn
Device(config-l2vpn)# redundancy predictive enabled
Device(config-l2vpn)# end
```

Example: Configuring Predictive Switchover (Xconnect Configuration Mode)

```
Device> enable
Device# configure terminal
Device(config)# l2vpn xconnect context con1
Device(config-xconnect)# redundancy predictive enabled
Device(config-xconnect)# end
```

Additional References

Related Documents

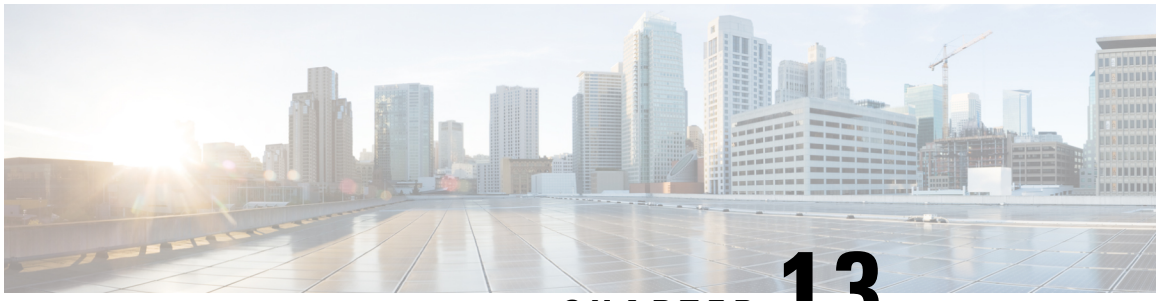
Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
MPLS commands	Cisco IOS Multiprotocol Label Switching Command Reference

Standards and RFCs

Standard/RFC	Title
RFC 4447	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html



CHAPTER 13

Configuring Routed Pseudowire and VPLS

Routed Pseudowire and VPLS feature routes Layer 3 traffic and Layer 2 frames for pseudowire connections between provider edge (PE) devices using Virtual Private LAN Services (VPLS) multipoint PE.

Prerequisites for Routed Pseudowire and VPLS

- MTU must be manually configured for MPLS enabled interfaces.

Restrictions for Routed Pseudowire and VPLS

- MPLS is *not* supported over routed VPLS in releases prior to Cisco IOS XE 16.6.1
- Maximum number of routed VPLS supported per system is 128.
- Maximum number of pseudowires supported per bridge domain is 62.
- Layer 2 and Layer 3 multicast are *not* supported.
- ACL on the core network is *not* supported.
- PBR is *not* supported.
- MTU check is *not* supported. MTU must be manually configured for MPLS enabled interfaces.

Restrictions on RSP3 Module

- VRRP and HSRP over VPLS BDI is *not* supported.

- Throughput is impacted as the packet is subjected to one extra pass for processing in both the imposition and the disposition flow.
- Multicast over routed pseudowire is *not* supported.
- Routed EoMPLS is *not* supported.
- FRR over routed pseudowire is *not* supported.
- BFD over routed pseudowire is *not* supported.
- MTU check is not performed on core facing interface. Same MTU has to be configured manually on all MPLS enabled interfaces in the network.
- IPv6 traffic is not supported over routed pseudowire.

Information About Routed Pseudowire and VPLS

Routed Pseudowire and VPLS

Routed Pseudowire and VPLS configuration can route Layer 3 traffic as well as Layer 2 frames for pseudowire connections between provider edge (PE) devices using Virtual Private LAN Services (VPLS) multipoint PE. The ability to route frames to and from these interfaces supports termination of pseudowires into the Layer 3 network (VPN or global) on the same switch, or to the tunnel Layer 3 frames over a Layer 2 tunnel (VPLS).

To configure routing support for a pseudowire, configure the IP address and other Layer 3 features for the Layer 3 domain in interface configuration mode.

Note BFD over BDI is supported with routed VPLS configuration.

Routed Pseudowire and VPLS on the RSP3 Module

Starting Cisco IOS Release 16.6.1, Routed pseudowire and VPLS is supported on the RSP3 module.

Routed VPLS is the ability to route and bridge frames to and from the pseudowires. Routed VPLS is configured by assigning the IP address under the bridge domain interface (BDI), and then associating that BDI with **l2 vfi mode** for VPLS. This feature combines the traditional Layer2 functionality with Layer3 routing functions.

Some of the benefits of Routed VPLS are:

- Offers new service opportunities such as virtual leased-line service and PVC-like layer-based service.
- Reduces cost by consolidating multiple core technologies into a single packet-based network infrastructure.
- Provides simplified services such as Layer2 transport options for service providers who need to provide L2 connectivity and maintain customer autonomy.
- Protects existing investments when networks extend their customer access to existing Layer2 networks without deploying a new separate infrastructure.

How to Configure Routed Pseudowire and VPLS

Configuring Routed Pseudowire andVPLS on the RSP3 Module

PE (RSP3) configuration

```
l2 vfi 102 manual
vpn id 102
bridge-domain 102
neighbor 3.3.3.3 encapsulation mpls
```

Access side interface

```
interface GigabitEthernet0/0/0
 no ip address
 load-interval 30
 negotiation auto
 service instance 1 ethernet
  encapsulation untagged
  bridge-domain 175

 service instance 2 ethernet
  encapsulation dot1q 102
  rewrite ingress tag pop 1 symmetric
  bridge-domain 102


interface BDI102
 ip address 188.0.0.1 255.255.0.0

interface BDI175
 ip address 175.0.0.1 255.255.0.0
```

Assigning IP Addresses For Bridge Domain (BDI)

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface bdi** *bdi-number* **Example:**	Configures the bridge domain interface.

	Command or Action	Purpose
	`Router(config)# interface bdi 3000`	
Step 4	**ip address** *ip address subnet mask* **Example:** `Router(config-if)# ip address 24.24.24.24 255.255.255.0`	Specifies the IP address for the bridge domain.
Step 5	**no shut** **Example:** `Router(config-if)# no shutdown`	Enables the bridge domain interface.
Step 6	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode.

Configuring a VFI on a PE Device

The virtual forwarding interface (VFI) specifies the VPN ID of a Virtual Private LAN Services (VPLS) domain, the addresses of other provider edge (PE) devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer. Perform this task to configure a VFI:

Note Only Multiprotocol Label Switching (MPLS) encapsulation is supported.

Note You must configure BDI on the bridge domain that has the association with the VFI.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *name* **manual** **Example:**	Establishes a Layer 2 VPN (L2VPN) virtual forwarding interface (VFI) between two or more separate networks and enters VFI configuration mode.

	Command or Action	Purpose
	`Device(config)# l2 vfi vfi110 manual`	
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 110`	Configures a VPN ID for a VPLS domain. • The emulated VCs bound to this Layer 2 virtual routing and forwarding (VRF) instance use this VPN ID for signaling.
Step 5	**neighbor** *remote-router-id vc-id* {**encapsulation** *encapsulation-type* | **pw-class** *pw-name*} [**no-split-horizon**] **Example:** `Device(config-vfi)# neighbor 172.16.10.2 4 encapsulation mpls`	Specifies the type of tunnel signaling and encapsulation mechanism for each VPLS peer. **Note** Split horizon is the default configuration to avoid broadcast packet looping and to isolate Layer 2 traffic. Use the **no-split-horizon** keyword to disable split horizon and to configure multiple VCs per spoke into the same VFI.
Step 6	**bridge-domain** *bd-id* **Example:** `Device(config-vfi)# bridge-domain 100`	Specifies a bridge domain.
Step 7	**end** **Example:** `Device(config-vfi)# end`	Exits VFI configuration mode and returns to privileged EXEC mode.

Configuration Examples: Routed Pseudowire and VPLS

Example: Configuring Routed Pseudowire and VPLS

The example configures the IP address on a BDI interface and associates the interface to a VFI.

```
!
interface GigabitEthernet0/0/0
 service instance 3 ethernet
  encapsulation dot1q 3000
  rewrite ingress tag pop 1 symmetric
  bridge-domain 100
!
interface BDI100
 ip address 24.24.24.24 255.255.255.0
 no shut
!
```

```
l2 vfi TEST manual
 vpn id 100
 bridge-domain 100
 neighbor 9.9.9.9 encapsulation mpls
!
```

Verifying the Configuration on the RSP3 Module

Use the follwing show commands to verify routed pseudowire and VPLS configurations on the RSP3.

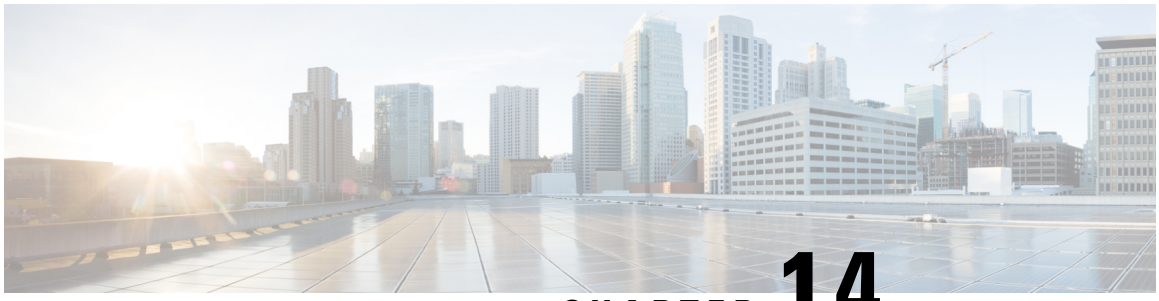
- **show l2vpn vfi d**
- **show mpls ldp bindings local-label**
- **show mpls forwarding-table**
- **show ip cef**
- **show platform ha pp act pw vpls**

show mpls l2transport vc

```
Router# show mpls l2transport vc 100

Local intf     Local circuit              Dest address    VC ID      Status
-------------  -------------------------     ---------------      ----------   ----------
Gi0/2          Eth VLAN 100               192.168.1.7         100          UP
ASR900#
```

show mpls l2transport summary

```
Router# show mpls l2transport summary
Destination address: 110.0.0.3, total number of vc: 226
  0 unknown, 0 up, 125 down, 101 admin down, 0 recovering, 0 standby, 0 hotstandby
  99 active vc on MPLS interface Gi0/16
```



CHAPTER 14

MPLS over Routed Pseudowire

Note This feature is supported only on the Cisco RSP3 Module.

Routed pseudowire provides the ability to route layer 3 in addition to the layer 2 bridge frames to and from pseudowire. Routed pseudowire is configured by assigning IP address under the bridge domain interface (BDI) in addition to the **vfi** command.

Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP), and Border Gateway Protocol (BGP) configurations are supported over routed pseudowire BDI.

Restrictions for MPLS over Routed Pseudowire

- IPv6 traffic is not supported over routed pseudowire.
- Loop Free Alternate/Remote Loop Free Alternate feature is not supported over routed pseudowire.
- Bidirectional Forwarding (BFD) is not supported over routed pseudowire.
- Precision Time Protocol (PTP) is not supported over routed pseudowire.
- QoS is not supported over routed pseudowire.
- Multicast is not supported over routed pseudowire.
- Virtual Router Redundancy Protocol (VRRP) and Hot Standby Redundancy Protocol (HSRP) is not supported over routed pseudowire.
- Access control lists (ACL) is not supported over routed pseudowire.

Configuring MPLS over Routed Pseudowire and VPLS

```
l2 vfi VPLS100 manual
 vpn id 100
 bridge-domain 100
 neighbor 4.4.4.4 encapsulation mpls
!

interface BDI100
 ip address 192.0.41.1 255.255.255.0
 ip ospf network point-to-point
 mpls ip
!
```

MPLS over Routed Pseudowire and BDI Configuration

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface bdi** *bdi-number* **Example:** `Router(config)# interface bdi 3000`	Configures the bridge domain interface.
Step 4	**ip address** *ip address subnet mask* **Example:** `Router(config-if)# ip address 209.165.201.10 255.255.255.224`	Specifies the IP address for the bridge domain.
Step 5	**mpls ip** **Example:** `Router(config-if)# mpls ip`	Enables MPLS support over bridge domain interface.
Step 6	**no shut** **Example:** `Router(config-if)# no shutdown`	Enables the bridge domain interface.

	Command or Action	Purpose
Step 7	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode.

Verify MPLS over Routed Pseudowire BDI Configuration

The following command shows the virtual circuit operational status:

```
router#show mpls l2transport vc

Local intf       Local circuit              Dest address    VC ID      Status
-------------  -------------------------- --------------- ---------- ----------
VFI VPLS100       vfi                          209.165.201.1        100       UP


R1-Act#show mpls l2transport summary
Destination address: 209.165.201.1, total number of vc: 1167
  0 unknown, 1167 up, 0 down, 0 admin down, 0 recovering, 0 standby, 0 hotstandby
1167 ive vc on MPLS interface Te0/2/8
```

The following command shows the virtual circuit details:

```
router#show mpls l2transport vc 100 detail
Local interface: VFI VPLS100 vfi up
  Interworking type is Ethernet
  Destination address: 209.165.201.1, VC ID: 100, VC status: up
    Output interface: Te0/2/8, imposed label stack {1204 794}
    Preferred path: not configured
    Default path: active
    Next hop: 209.165.201.10
  Create time: 1d17h, last status change time: 1d17h
    Last label FSM state change time: 1d17h
  Signaling protocol: LDP, peer 209.165.201.1:0 up
    Targeted Hello: 209.165.201.2(LDP Id) -> 209.165.201.1, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: configured and enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 2352, remote 794
    Group ID: local n/a, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: On (configured: autosense)
  SSO Descriptor: 209.165.201.1/100, local label: 2352
  Dataplane:
    SSM segment/switch IDs: 25556223/16931917 (used), PWID: 2337
```

```
  VC statistics:
    transit packet totals: receive 0, send 0
    transit byte totals:   receive 0, send 0
    transit packet drops:  receive 0, seq error 0, send 0
```

The following command shows the virtual forwarding instance details:

```
router#show l2vpn vfi detail
Legend: RT=Route-target, S=Split-horizon, Y=Yes, N=No

VFI name: VPLS100, state: up, type: multipoint, signaling: LDP
  VPN ID: 100
  Bridge-Domain 100 attachment circuits:
  Pseudo-port interface: pseudowire100001
  Interface          Peer Address     VC ID        S
  pseudowire100002   209.165.201.1         100         Y
```

The following command shows the MPLS LDP neighbor details:

```
router#show mpls ldp neighbor
    Peer LDP Ident: 209.165.201.1:0; Local LDP Ident 209.165.201.2:0
        TCP connection: 209.165.201.1.26053 - 209.165.201.2.646
        State: Oper; Msgs sent/rcvd: 7022/5737; Downstream
        Up time: 2d01h
        LDP discovery sources:
          TenGigabitEthernet0/2/8, Src IP addr: 209.165.201.5
          Targeted Hello 209.165.201.2 -> 209.165.201.6, active, passive
        Addresses bound to peer LDP Ident:
          209.165.201.1   209.165.201.3     209.165.201.4      209.165.201.5
          192.0.45.2      192.0.43.2      192.0.49.2      192.0.50.2
          192.0.56.2      192.0.55.2      192.0.62.2      192.0.48.2
          192.0.61.2      192.0.41.2      192.0.46.2      192.0.52.2
          192.0.63.2      192.0.60.2      192.0.57.2      192.0.58.2
          192.0.64.2      192.0.47.2      192.0.54.2      192.0.59.2
          192.0.51.2      192.0.42.2      192.0.65.2      192.0.44.2
          192.0.53.2
```

The following command shows the label allocated by the LDP protocol:

```
router#show mpls ldp bindings
  lib entry: 209.165.201.7/24, rev 27617
        local binding:  label: imp-null
        remote binding: lsr: 209.165.201.1:0, label: imp-null
```

The following command shows the LFIB entries:

```
router#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
3543       1206       210.210.210.0/24 0             BD2778     209.165.201.10
```

The following command shows the PI CEF chain:

```
router#show ip cef 209.165.202.129 internal
209.165.201.7/24, epoch 3, flags [att, cnn, cover, deagg], RIB[C], refcnt 6, per-destination
 sharing
  sources: RIB
  feature space:
    IPRM: 0x0003800C
    Broker: linked, distributed at 2nd priority
    LFD:  209.165.201.7/24 0 local labels
        contains path extension list
  subblocks:
```

```
    gsb Connected chain head(1): 0x4E597FA0
    Covered dependent prefixes: 2
      need deagg: 2
  ifnums:
    BDI100(37)
  path list 4DF8B8A8, 3 locks, per-destination, flags 0x49 [shble, rif, hwcn]
    path 4DC54A08, share 1/1, type connected prefix, for IPv4
      MPLS short path extensions: [none] MOI flags = 0x1 label implicit-null
      connected to BDI100, glean
  output chain:
    glean
```

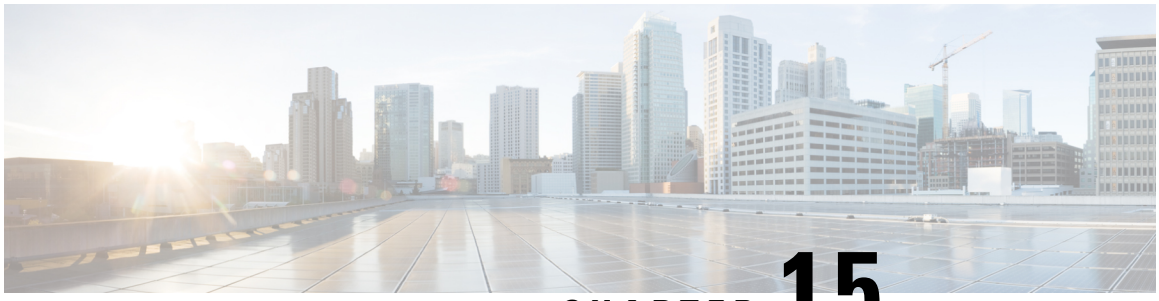
The following command shows the hardware programming for VPLS pseudowire:

```
router#show platform hardware pp active pw vpls
pw             : VFI557085844    bdomain         : 100        vsi       : 0x3
peer_ip        : 209.165.201.1       vc_id           : 100        has_cw    : 1
STP            : FWD             status          : Enabled     sh_group  : 0
local_label    : 17              remote_label    : 16          sh_type   : Hub
imp_oce        : 0x28116624      disp_oce        : 0x25D9A014  label_oce : 0x288241CC
pwe_lif        : 0x95FE          psn_fec         : 0x200004B1  encap_id  : 0x95FE
dest_gport     : 0x6C0000D1      ing_gport       : 0x189095FE  egr_gport : 0x18A095FE
imp_flow_label : No              disp_flow_label : No
```

The following command shows the BGP neighbourship status:

```
router#show ip bgp summary
BGP router identifier 13.13.13.13, local AS number 100
BGP table version is 9, main routing table version 9
8 network entries using 1152 bytes of memory
11 path entries using 968 bytes of memory
6/6 BGP path/bestpath attribute entries using 1008 bytes of memory
4 BGP rrinfo entries using 96 bytes of memory
2 BGP community entries using 48 bytes of memory
34 BGP extended community entries using 1360 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 4632 total bytes of memory
BGP activity 42/0 prefixes, 239/0 paths, scan interval 60 secs

Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
209.165.202.129      4          100    1117    1122        9    0    0 16:49:00        0
```

Note With scaled up OSPF interfaces over router pseudowire, there is a possibility of OSPF PDU size going beyond egress interface default MTU, causing instability in OSPF adjacency. Hence, it is recommended to have higher egress interface MTU (> 1540 byte) over which the above sessions are created.



CHAPTER 15

VPLS over Backup Pseudowire

Pseudowire redundancy allows you to detect any failure in the network and reroute the Layer 2 service to another endpoint. The other endpoint can continue to provide this service by providing additional backup pseudowire. This feature provides the ability to recover from a failure of either the remote provider edge (PE) router or the link between the PE and customer edge (CE) routers.

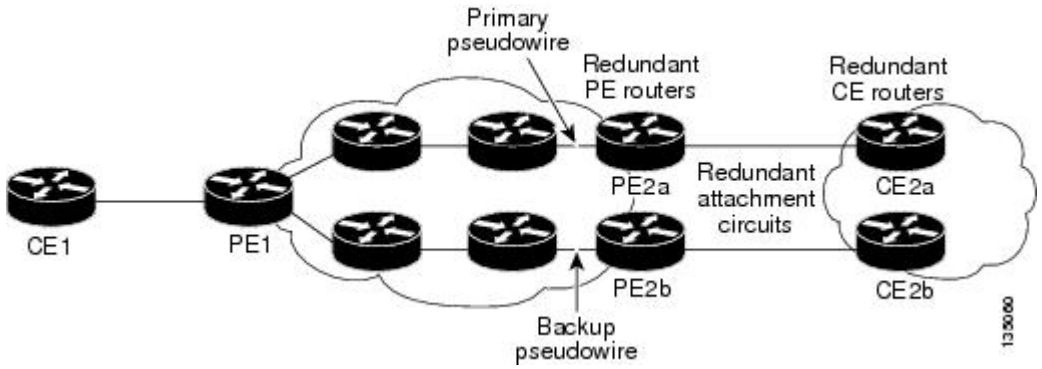
L2VPNs can provide pseudowire resiliency through their routing protocols. When the connectivity between the end-to-end PE routers fails, an alternative path to the directed LDP session and the user data takes over. However, there are some parts of the network in which this rerouting mechanism does not protect against interruptions in service.

RSP3 implementation reuses the EoMPLS configuration to achieve VPLS over the backup pseudowire functionality. The expected convergence in virtual circuit switchover and fallback is approximately 200-300 ms, which increases with scale configuration because only active pseudowire is programmed in hardware.

These are some scenarios in which pseudowire switchover takes place:

- Core link flap
- Access link flap
- Default access interface
- Remote loopback interface flap
- LDP disable or enable
- Control Word change
- VPN ID change within pseudowire configuration
- Cable pull
- Core side interface module reload
- Core side BFD flap (if configured)
- Core side IGP down
- Remote PE or P node reload or crash

Figure 15: Sample Topology for VPLS over Backup Pseudowire Deployment Scenario



Prerequisites for VPLS over Backup Pseudowire

- IGP and LDP should be up and running between peer devices.
- MTU must be manually configured for MPLS enabled interfaces.

Restrictions for VPLS over Backup Pseudowire

- Only one active and one backup pseudowire (PW) for each bridge domain (BD) is supported.
- Hierarchal Virtual Private Lan Service (HVPLS) is not supported.
- VPLS with BGP auto discovery is not supported.
- 1000 active PW with one backup PW for each VPLS session is supported.
- This feature is supported only with the new VPLS configuration model (on all the PE nodes).
- VFIs configured with the old configuration model cannot coexist with the VFIs of the new configuration model on the same BD.
- The member BDI should not be in the same group as the PW under l2vpn xconnect context configuration.
- More than two PW should not be configured in L2VPN context.
- In an L2VPN cross-connect context, the member BDI and the member physical interface cannot be configured at the same time. It leads to error objects.
- The maximum scale for VPLS session is 1000.
- The BDI used for this feature should not be configured with any IP configuration like MPLS IP, DHCP IP,or static IP.

- Routed PW cannot be configured because BDI interface cannot be configured with IP address.

Convergence Time for the VPLS Sessions

Table 10: Convergence Time (in Milliseconds) with One VPLS Session

Packet Size (Bytes)	Convergence Time (approximate)	Type Of Packet
64	213	IP
128	186	IP
256	173	IP
512	170	IP
1028	186	IP
1400	167	IP

Table 11: Convergence Time (in Milliseconds) with Multiple VPLS Session

Number of Virtual Circuits	Scenario	Convergence Time (approximate)
10	Active Down	373.56
	Reoptimization	15.73
100	Active Down	1880
	Reoptimization	517.93

VPLS over Backup Pseudowire Configuration

The following example shows the configuration on provider edge (PE) router 1 having two links to PE router 2 with IP address 209.165.200.225 and PE router 3 with IP address 209.165.200.226:

Configuration on PE router 1:

```
interface GigabitEthernet0/2/3
 no ip address
   negotiation auto
 service instance 1 ethernet
    encapsulation dot1q 1
    rewrite ingress tag pop 1 symmetric
!
bridge-domain 1000
 member GigabitEthernet0/2/3 service-instance 1
!
interface pseudowire10
 encapsulation mpls
 neighbor 209.165.200.225 1000
!
```

```
interface pseudowire20
 encapsulation mpls
 neighbor 209.165.200.226 2000
!
interface BDI1000
 no ip address
!
l2vpn xconnect context VC_1
 member BDI1000
 member pseudowire10 group Grp_1 priority 1
 member pseudowire20 group Grp_1 priority 2
```

Configuration on PE router 2:

```
interface GigabitEthernet0/2/4
 no ip address
 carrier-delay msec 0
 negotiation auto
 service instance 1 ethernet
    encapsulation dot1q 1
    rewrite ingress tag pop 1 symmetric!
bridge-domain 1
 member GigabitEthernet0/2/4 service-instance 1
!
interface pseudowire10
 encapsulation mpls
 neighbor 209.165.200.224 10
!
interface BDI1
 no ip address
!
l2vpn xconnect context VC_1
 member BDI1
 member pseudowire10 group Grp_1 priority 1
```

Configuration on PE router 3:

```
interface GigabitEthernet0/2/0
 no ip address
 carrier-delay msec 0
  negotiation auto
  service instance 1 ethernet
    encapsulation dot1q 1
    rewrite ingress tag pop 1 symmetric
!
bridge-domain 1
 member GigabitEthernet0/2/0 service-instance 1
!
interface pseudowire10
 encapsulation mpls
 neighbor 209.165.200.224 20
!
interface BDI1
 no ip address
!
l2vpn xconnect context VC_1
 member BDI1
 member pseudowire10 group Grp_1 priority 1
```

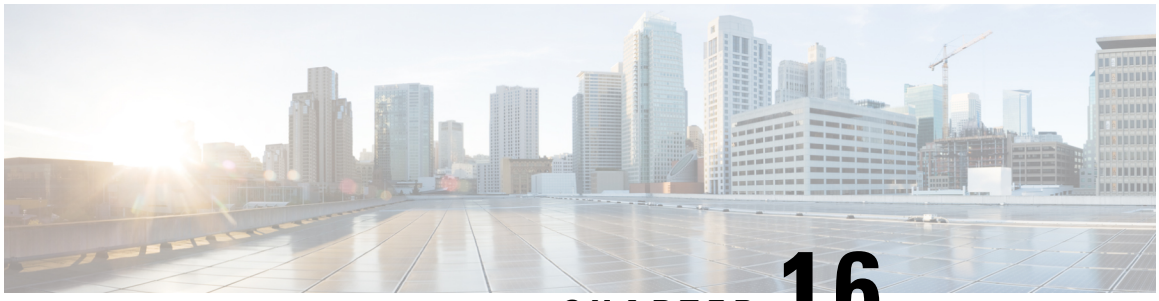
Verify VPLS over Backup Pseudowire Configuration

Use the following commands to verify the VPLS over backup pseudowire configuration on the PE router 1 head node:

```
PE1#show mpls l2transport vc
Local intf     Local circuit              Dest address    VC ID      Status
-------------  -------------------------- --------------- ---------- ----------
BD1000         Eth VLAN 1000              209.165.200.225    1000       UP
BD1000         Eth VLAN 1000              209.165.200.226    2000       STANDBY


PE1#show mpls l2transport vc 1000 detail
Local interface: BD1000 up, line protocol up, Eth VLAN 1000 up
  Interworking type is Ethernet
  Destination address: 209.165.200.225, VC ID: 1000, VC status: up
    Output interface: Gi0/2/5, imposed label stack {24 24}
    Preferred path: not configured
    Default path: active
    Next hop: 192.168.1.2
  Create time: 00:08:46, last status change time: 00:07:14
    Last label FSM state change time: 00:07:12
  Signaling protocol: LDP, peer 209.165.200.225:0 up
    Targeted Hello: 209.165.200.227(LDP Id) -> 209.165.200.225, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: configured and enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LruRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: No fault
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: No fault
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
    MPLS VC labels: local 24, remote 24
    Group ID: local n/a, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: On (configured: autosense)
  SSO Descriptor: 209.165.200.225/1000, local label: 24
  Dataplane:
    SSM segment/switch IDs: 4186/4180 (used), PWID: 6
  VC statistics:
    transit packet totals: receive 0, send 0
    transit byte totals:   receive 0, send 0
    transit packet drops:  receive 0, seq error 0, send 0


PE1#show mpls l2transport vc 2000 detail
Local interface: BD1000 up, line protocol up, Eth VLAN 1000 up
  Interworking type is Ethernet
  Destination address: 209.165.200.226, VC ID: 2000, VC status: standby
    Output interface: Gi0/2/1, imposed label stack {21 16}
    Preferred path: not configured
    Default path: active
    Next hop: 192.168.3.2
  Create time: 00:08:51, last status change time: 00:08:51
```

```
    Last label FSM state change time: 00:05:40
  Signaling protocol: LDP, peer 209.165.200.226:0 up
    Targeted Hello: 209.165.200.227(LDP Id) -> 209.165.200.226, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: configured and enabled
    Status TLV support (local/remote)   : enabled/supported
      LDP route watch                   : enabled
      Label/status state machine        : established, LrdRru
      Last local dataplane   status rcvd: No fault
      Last BFD dataplane     status rcvd: Not sent
      Last BFD peer monitor  status rcvd: No fault
      Last local AC  circuit status rcvd: DOWN(standby)
      Last local AC  circuit status sent: No fault
      Last local PW i/f circ status rcvd: No fault
      Last local LDP TLV     status sent: DOWN(standby)
      Last remote LDP TLV    status rcvd: No fault
      Last remote LDP ADJ    status rcvd: No fault
      MPLS VC labels: local 25, remote 16
    Group ID: local n/a, remote 0
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  Control Word: On (configured: autosense)
  SSO Descriptor: 209.165.200.226/2000, local label: 25
  Dataplane:
    SSM segment/switch IDs: 12382/8277 (used), PWID: 7
  VC statistics:
    transit packet totals: receive 0, send 0
    transit byte totals:   receive 0, send 0
    transit packet drops:  receive 0, seq error 0, send 0


PE1#show bridge-domain 1000
Bridge-domain 1000 (3 ports in all)
State: UP                    Mac learning: Enabled
Aging-Timer: 300 second(s)
Maximum address limit: 65534
    BDI1000  (up)
    GigabitEthernet0/2/3 service instance 1
    BDI1000 service instance 1


PE1#show l2vpn service all detail
Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down             AD=Admin Down     IA=Inactive
        SB=Standby  HS=Hot Standby      RV=Recovering     NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                   Prio  St  XC St
  ---------          -----       -------------                   ----  --  -----
VPWS name: VC_1, State: UP
  pw1000             Grp_1       209.165.200.225:1000(MPLS)              1     UP  UP
                                 Local VC label 24
                                 Remote VC label 24

  pw2000             Grp_1       209.165.200.226:2000(MPLS)              2     SB  IA
                                 Local VC label 25
                                 Remote VC label 16

  BD1000                         BD1000:1000(Eth VLAN)           0     UP  UP
                                 Interworking: ethernet


PE1#show l2vpn service all
```

```
Legend: St=State    XC St=State in the L2VPN Service      Prio=Priority
        UP=Up       DN=Down            AD=Admin Down      IA=Inactive
        SB=Standby  HS=Hot Standby     RV=Recovering      NH=No Hardware
        m=manually selected

  Interface          Group       Encapsulation                  Prio  St  XC St
  ---------          -----       -------------                  ----  --  -----
VPWS name: VC_1, State: UP
  pw1000             Grp_1       209.165.200.225:1000(MPLS)            1     UP  UP
  pw2000             Grp_1       209.165.200.226:2000(MPLS)            2     SB  IA
  BD1000                         BD1000:1000(Eth VLAN)          0     UP  UP
```



CHAPTER 16

EVPN Single-Homing Over MPLS

The EVPN Single-Homing feature utilizes the functionality defined in RFC 7432 (BGP MPLS-based Ethernet VPN), to achieve single-homing between a Provider Edge (PE) and a Customer Edge (CE) device.

Feature History

Table 12: Feature History

Feature Name	Release Information	Feature Description
EVPN Single-Homing Over MPLS for NCS 4201 and NCS 4202	Cisco IOS XE Amsterdam 17.3.1	The EVPN Single-Homing feature utilizes the BGP MPLS-based Ethernet VPN functionality as defined in RFC 7432. That is, to achieve single-homing between a Provider Edge (PE) and a Customer Edge (CE) device. There are three fundamental building blocks for EVPN technology, EVPN Instance (EVI), Ethernet Segment (ES), EVPN BGP routes and extended communities. For EVPN Single-Homing feature, a CE device is attached to a single PE device and has an Ethernet Segment.

Feature Name	Release Information	Feature Description
EVPN Single-Homing Over MPLS for NCS 4206 and NCS 4216	Cisco IOS XE Amsterdam 17.1.1	The EVPN Single-Homing feature utilizes the BGP MPLS-based Ethernet VPN functionality as defined in RFC 7432. That is, to achieve single-homing between a Provider Edge (PE) and a Customer Edge (CE) device. There are three fundamental building blocks for EVPN technology, EVPN Instance (EVI), Ethernet Segment (ES), EVPN BGP routes and extended communities. For EVPN Single-Homing feature, a CE device is attached to a single PE device and has an Ethernet Segment.

Information about EVPN Single-Homing

Ethernet Multipoint Connectivity

To achieve Ethernet multipoint connectivity, MPLS deployments traditionally rely on Virtual Private LAN Services (VPLS). A VPLS service is built with a full-mesh of pseudowires between PE devices that are part of a Layer 2 broadcast domain. A VPLS PE device performs data-plane MAC learning. For MAC learning, the VPLS PE device uses local interfaces for traffic coming from the access network and uses pseudowires for the traffic coming from the core network.

EVPN Multipoint Solution

EVPN is the next generation of multipoint L2VPN solution that aligns operation principles of L3VPN with Ethernet services. Instead of relying solely on data plane for MAC Address learning, EVPN PE devices signal and learn MAC addresses over the core network using BGP, while still using data plane MAC-learning on the access side. Providers can configure BGP as a common VPN control plane for their ethernet offerings and leverage the advantages of Layer 3 VPN over VPLS. In Cisco IOS XE Fuji 16.8.1, only Single Homing functionality is supported from the feature set defined in RFC 7432.

EVPN Building Blocks

There are three fundamental building blocks for EVPN technology, EVPN Instance (EVI), Ethernet Segment (ES), EVPN BGP routes and extended communities:

- EVI is a VPN connection on a PE router. It is the equivalent of IP VPN Routing and Forwarding (VRF) in Layer 3 VPN. It is also known as MAC-VRF.

- ES is a connection with a customer site (device or network) and is associated with access-facing interfaces. Access-facing interfaces are assigned unique IDs that are referred to as Ethernet Segment Identifiers (ESI). A site can be connected to one or more PEs. The ES connection has the same ESI in each PE connected to the site.
- RFC 7432 defines routes and extended communities to enable EPVN support. In Cisco IOS XE Fuji 16.8.x Software Release, Route Type 2 and Route Type 3 are supported.

  In BGP MPLS-based EVPN, an EVI is configured for every PE device for each customer associated with the PE device. In this case, a customer is any customer edge device that is attached to the PE device. The CE device can be a host, a switch or a router. Each EVI has a unique Route Distinguisher (RD) and one or more Route Targets (RT).

  For EVPN Single-Homing feature, a CE device is attached to a single PE device and has an Ethernet Segment with ESI=0.

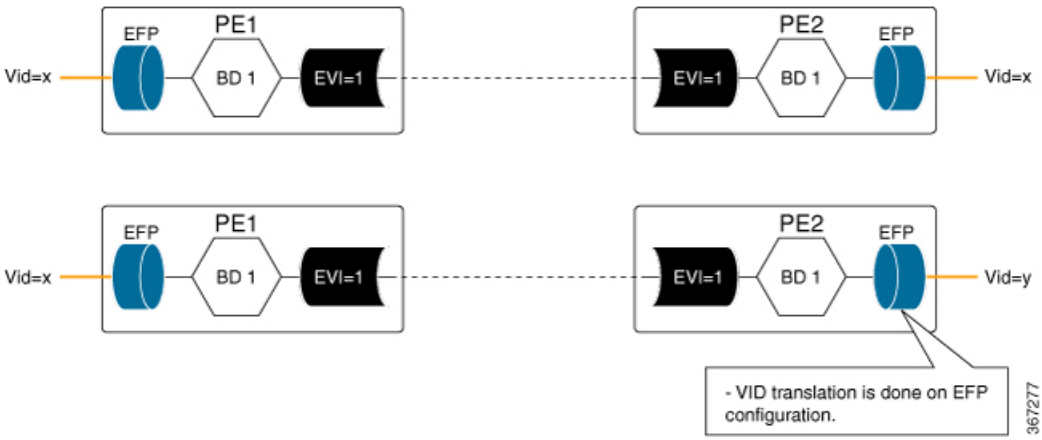
Service Interfaces

The following are types of EVPN VLAN service interfaces:

VLAN-based Service Interface

In VLAN-based service interface, each VLAN is associated to one bridge domain and one EVI.

Figure 16: VLAN-Based Service Interface



For VLAN-based Service Interface, Type 1 Route Distinguisher, a unique number used to distinguish identical routes in different VRFs, is used for EVIs as recommended by the RFC 7432. The Route Distinguishers and Router Targets, which are used to share routes between different VRFs, are autogenerated to ensure unique Route Distinguisher numbers across EVIs.

VLAN Bundle Service Interface

In VLAN Bundle Service Interface, multiple VLANs share the same bridge table.

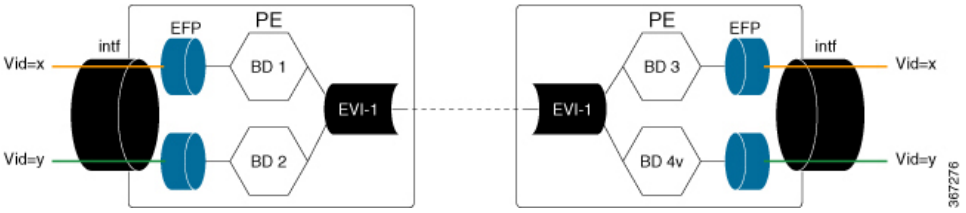
Figure 17: VLAN Bundle Service Interface

Each EVPN instance corresponds to multiple broadcast domains maintained in a single bridge table per MAC-VRF. For VLAN Bundle Service Interface service to work, MAC addresses must be unique across all VLANs for an EVI.

VLAN-Aware Bundle Service Interface

For VLAN-aware Bundle Service Interface, each VLAN is associated with one bridge domain, but there can be multiple bridge domains associated with one EVI.

Figure 18: VLAN-Aware Bundle Service Interface



An EVPN instance consists of multiple broadcast domains where each VLAN has one bridge table. Multiple bridge tables (one per VLAN) are maintained by a single MAC-VRF that corresponds to the EVPN instance.

Route Types

For EVPN Single-Homing feature, Route Type 2 and Route Type 3 are supported, as defined by RFC 7432.

Route Type 2 — MAC and IP Advertisement Route

Type 2 Routes are used to advertise MAC addresses and their associated IP addresses. When a PE router learns the MAC address of a CE device that is connected to it locally, or a MAC address of a device behind the CE device, a MAC and an IP advertisement route is created.

The following table describes the header format for the MAC and IP Advertisement Route packet:

Table 13: Header format for the MAC and IP Advertisement Route packet

Field	Value	Length (Octets)
Route Type	0x02	1
Length	Variable	1
EVI RD	Type 1 (IPv4 address) RD unique across all EVIs on the PE	8
ESI	Ethernet Segment Identifier	10
Ethernet Tag	0 or valid Ethernet Tag	4

Field	Value	Length (Octets)
MAC Addr Len	48	1
MAC Address	Valid MAC address	6
IP Addr Length	IP address length in bits: 0, 32 or 128	1
IP Address	Optional IP address	0 or 4 or 16
Label1	Valid downstream assigned label to perform forwarding to a CE device based on the destination MAC address	3
Label2	Specifies a second label	0-3
EVI RT	Type 0 (2byteAS) route target	8

Note

- MAC Address field is populated with the CE address.
- IP address field is optional with IP Address length set to 0 bits.
- For EVPN Single-Homing feature, ESI value is always set to 0.
- In the Label field (Label1, Label2), Per-BD or Per-CE labels can be assigned.
  - Per-BD is used when PE advertises a single label for all MAC addresses learned in a given bridge domain.
  - Per-CE label assigns a separate label to each access port in the bridge domain.

Route Type 3 — Inclusive Multicast Ethernet Tag Route

Type 3 routes are used for transporting Broadcast, Unknown Unicast, and Multicast (BUM) traffic to other PE devices across a given EVPN network instance.

The following tables describes the header format for Type 3 routes:

Table 14: Header Format for Type 3 Route Packets

Field	Value	Length (Octets)
Route Type	0x03	1
Length	26 or 38	1
EVI RD	Type 1 (IPv4Addr) RD unique across all EVIs on the PE	8
Ethernet Tag	0 or valid Ethernet Tag	4

Field	Value	Length (Octets)
IP Addr Length	IP Address Length - 32 bits or 128 bits	1
IP Address	IP Address common for all EVIs (for example, loopback address)	4 or 16
PMSI Tunnel Attr	{1 byte flags = 0}:{1 byte Tunnel Type}:{3 byte label}:{variable length Tunnel Identifier}	Variable
EVI RT	Type 0 (2byteAS) route target	8

The PE devices advertise an Inclusive Multicast Ethernet Tag (IMET) Route for every EVI-Ethernet Tag sequence. The Ethernet Tag is set to 0 for VLAN-based and VLAN-bundling service interfaces. The Ethernet Tag is set to a valid VLAN ID for VLAN-aware bundling service interface.

Type 3 route also carries a Provider Multicast Service Interface (PMSI) Tunnel attribute as specified in RFC 6514 (BGP Encodings and Procedures for MVPNs).

For Ingress Replication, the IMET route is used to advertise the label (in the PMSI Tunnel Attribute) that the other PEs can use to send BUM traffic to the originating PE device.

Prerequisites for EVPN Single-Homing

- EVI and Bridge domains must be in established state with associated MPLS labels.

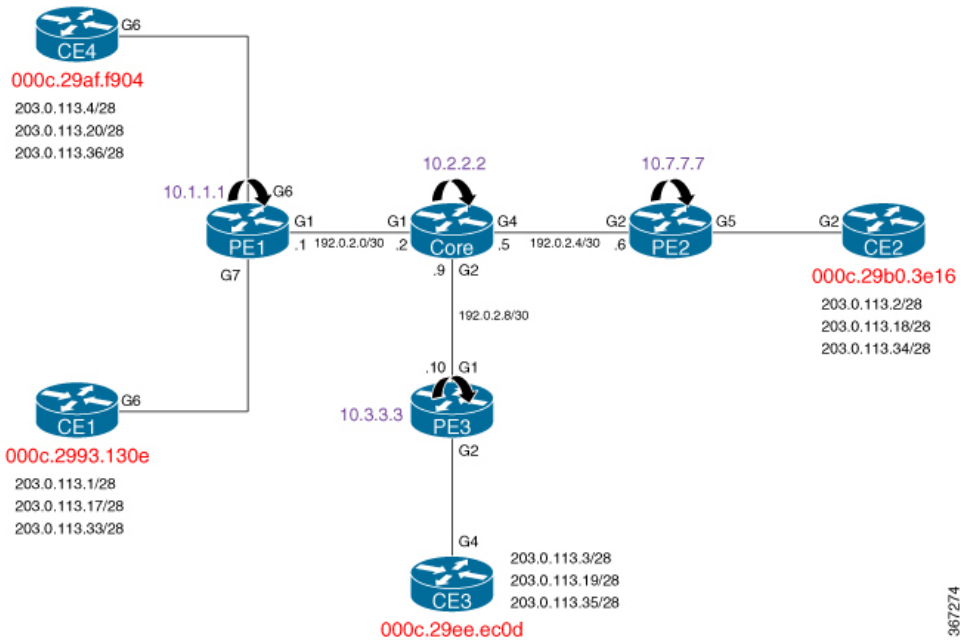
Restrictions for EVPN Single-Homing

- Route Type 1 and Route Type 4 are not supported.
- Per-EVI-based labelling is not supported.
- Maximum number of supported bridge domains is 1600.
- Maximum number of supported EEPs or service instances is 8000.
- Single-Homing feature is not supported with port channel interface between Provider Edge and Customer Edge devices.
- If want to create a VLAN-bundle or VLAN-aware EVI's, they must be configured before adding to a bridge domain (or VLAN).
- ESI must be all 0s.

How to Configure EVPN Single Homing

Configuring EVPN

Figure 19: EVPN Single Homing



The above figure represents a simple EVPN network. Use the following steps to configure EVPN:

EVPN Configuration

```
enable
    configure terminal
        l2vpn evpn
          replication-type ingress
          router-id Loopback1
          mpls label mode per-ce
         !
        l2vpn evpn instance 10 vlan-based
          route-distinguisher 10.1.1.1:10
          route-target both 10:10
          no auto-route-target
         !
        bridge-domain 10
        member evpn-instance 10
          member GigabitEthernet 0/0/1 service-instance 10
        !
        interface GigabitEthernet 0/0/1
          no ip address
          service instance 10 ethernet
           encapsulation dot1q 200
          ¡
         ¡
        !
```

Note In the above example, the **l2vpn evpn instance** command and the associated sub-mode is only required if one or more of the following apply:

- There is per-EVI configuration to be applied (for example, route targets or route distinguished)
- The EVI is VLAN-bundle or VLAN-aware.
- Configure member EVPN instance EVI under the bridge-domain without configuring EVPN instance.

If the EVPN instance is not explicitly configured, it is created automatically as a VLAN-based EVI with autogenerated route targets and route distinguisher.

Configuring L2VPN EVPN Globally and EVI on IOS-XE Router

```
l2vpn evpn
 replication-type ingress    ----> Enables ingress replication label
!
l2vpn evpn instance 10 vlan-based ---> Configures Vlan-based EVI 10
!
l2vpn evpn instance 20 vlan-bundle ---→ Configures Vlan-bundled EVI 20
!
l2vpn evpn instance 30 vlan-aware   ---→ Configures Vlan-aware EVI 30
```

Configuring Bridge Domains on IOS-XE Router

```
bridge-domain 10
 mac aging-time 30
 member GigabitEthernet6 service-instance 10  --→ Links SI 10 on interface with Bridge-domain
 10
 member evpn-instance 10 --→ Links EVI 10 with Bridge-domain 10
!
bridge-domain 20
 mac aging-time 30
 member GigabitEthernet6 service-instance 20 --→ Links SI 20 on interface with Bridge-domain
 20
 member evpn-instance 20 --> Links EVI 20 with Bridge-domain 20
!
bridge-domain 30
 mac aging-time 30
 member GigabitEthernet6 service-instance 30 --→ Links SI 30 on interface with Bridge-domain
 30
 member evpn-instance 30 ethernet-tag 30 ---> Links EVI 30 with Bridge-domain 30
```

Configuring Access Interface on a Provider Edge

```
interface GigabitEthernet6
 no ip address
 negotiation auto
 service instance 10 ethernet ----> Enables service instance 10 under the physical interface

  encapsulation dot1q 10
 !
 service instance 20 ethernet ----> Enables service instance 20 under the physical interface

  encapsulation dot1q 20-21
 !
 service instance 30 ethernet ----> Enables service instance 30 under the physical interface
```

```
 encapsulation dot1q 30
```

Configuring EVPN Single-Homing

Use the following steps to configure EVPN Single-Homing:

Configuring BGP on Provider Edge Device, PE1

```
enable
configure terminal
 router bgp 100
  bgp router-id 10.1.1.1
  bgp log-neighbor-changes
  bgp graceful-restart
  neighbor 10.2.2.2 remote-as 100
  neighbor 10.2.2.2 update-source Loopback0
 !
 address-family ipv4
  neighbor 10.2.2.2 activate
 exit-address-family
 !
 address-family l2vpn evpn     ----> Enables L2VPN EVPN address family
  neighbor 10.2.2.2 activate
  neighbor 10.2.2.2 send-community both
  neighbor 10.2.2.2 soft-reconfiguration inbound
 exit-address-family
```

Configuring BGP on Route Reflector

```
router bgp 100
 bgp router-id 10.2.2.2
 bgp log-neighbor-changes
 bgp graceful-restart
 neighbor 10.1.1.1 remote-as 100
 neighbor 10.1.1.1 update-source Loopback0
 neighbor 10.3.3.3 remote-as 100
 neighbor 10.3.3.3 update-source Loopback0
 neighbor 10.7.7.7 remote-as 100
 neighbor 10.7.7.7 update-source Loopback0
 !
 address-family ipv4
  neighbor 10.1.1.1 activate
  neighbor 10.1.1.1 route-reflector-client
  neighbor 10.3.3.3 activate
  neighbor 10.3.3.3 route-reflector-client
  neighbor 10.7.7.7 activate
  neighbor 10.7.7.7 route-reflector-client
 exit-address-family
 !
 address-family l2vpn evpn      ----> Enables L2vpn evpn address family
  neighbor 10.1.1.1 activate
  neighbor 10.1.1.1 send-community both
  neighbor 10.1.1.1 route-reflector-client
  neighbor 10.1.1.1 soft-reconfiguration inbound
  neighbor 10.3.3.3 activate
  neighbor 10.3.3.3 send-community both
  neighbor 10.3.3.3 route-reflector-client
  neighbor 10.3.3.3 soft-reconfiguration inbound
  neighbor 10.7.7.7 activate
  neighbor 10.7.7.7 send-community both
```

```
  neighbor 10.7.7.7 route-reflector-client
  neighbor 10.7.7.7 soft-reconfiguration inbound
 exit-address-family
```

Configuring Customer Edge and Provider Edge Interfaces

CE1 configuration

```
interface GigabitEthernet6.10
 encapsulation dot1Q 10
 ip address 203.0.113.1 255.255.255.240
interface GigabitEthernet6.20
 encapsulation dot1Q 20
 ip address 203.0.113.17 255.255.255.240
interface GigabitEthernet6.30
 encapsulation dot1Q 30
 ip address 203.0.113.33 255.255.255.240
```

PE1 Configuration

```
interface GigabitEthernet6
 no ip address
 negotiation auto
 service instance 10 ethernet
  encapsulation dot1q 10
 !
 service instance 20 ethernet
  encapsulation dot1q 20-21
 !
 service instance 30 ethernet
  encapsulation dot1q 30
```

Verification Examples for EVPN Single-Homing

Use the following command to verify that EVI and Bridge domains are in established state and to display associated MPLS labels:

```
show l2vpn evpn evi detail
EVPN instance:    10 (VLAN Based)    ----> VLAN Based EVI
  RD:             10.1.1.1:10 (auto) ----> RD derived from Loopback0 of PE1
  Import-RTs:     100:10
  Export-RTs:     100:10
  Per-EVI Label:  none
  State:          Established    ----> EVI state
  Encapsulation:  mpls
  Bridge Domain:  10
    Ethernet-Tag: 0
    BUM Label:    23             ----> Broadcast/Unknown unicast/Multicast traffic label
    Per-BD Label: 22
    State:        Established     ----> Bridge-domain state
    Pseudoports:
      GigabitEthernet6 service instance 10 ----> Local interface part of bridge-domain
      GigabitEthernet7 service instance 10   ---→ Local interface part of bridge-domain

EVPN instance:    20 (VLAN Bundle) ----> VLAN Bundled EVI
  RD:             10.1.1.1:20 (auto)
  Import-RTs:     100:20
  Export-RTs:     100:20
  Per-EVI Label:  none
  State:          Established
```

```
  Encapsulation:  mpls
  Bridge Domain:  20
    Ethernet-Tag: 0
    BUM Label:    20
    Per-BD Label: 21
    State:        Established
    Pseudoports:
      GigabitEthernet6 service instance 20
      GigabitEthernet7 service instance 20

EVPN instance:    30 (VLAN Aware)  ----> VLAN-Aware EVI
  RD:             10.1.1.1:30 (auto)
  Import-RTs:     100:30
  Export-RTs:     100:30
  Per-EVI Label:  none
  State:          Established
  Encapsulation:  mpls
  Bridge Domain:  30
    Ethernet-Tag: 30
    BUM Label:    18
    Per-BD Label: 19
    State:        Established
    Pseudoports:
      GigabitEthernet6 service instance 30
      GigabitEthernet7 service instance 30
```

Use the following command to verify that the bridge domain has learnt the local and remote MAC addresses:

```
PE1#show bridge-domain 10
Bridge-domain 10 (3 ports in all)
State: UP                    Mac learning: Enabled
Aging-Timer: 30 second(s)  ----> MAC aging timer for bridge-domain
    GigabitEthernet6 service instance 10
    GigabitEthernet7 service instance 10
    EVPN Instance 10
   AED MAC address    Policy  Tag       Age  Pseudoport
   -   000C.29B0.3E16 forward static_r  0    OCE_PTR:0xe8eb04a0 ---→ Remotely learnt MAC
   -   000C.29AF.F904 forward dynamic_c 29   GigabitEthernet6.EFP10 --→ MAC locally learnt

   -   000C.2993.130E forward dynamic_c 26   GigabitEthernet7.EFP10
   -   000C.29EE.EC0D forward static_r  0    OCE_PTR:0xe8eb0500
```

Note In the above output, MAC addresses with forward dynamic_c tags are locally learned addresses and MAC addresses with forward static_r tags are remote addresses learned through EVPN.

Use the following command to verify that EVPN manager has received the local MACs learned by the bridge domain:

```
PE1# show l2vpn evpn mac
MAC Address    EVI   BD    ESI                       Ether Tag  Next Hop
-------------- ----- ----- ------------------------- ---------- ---------------
000c.2993.130e 10    10    0000.0000.0000.0000.0000  0          Gi7:10
000c.29af.f904 10    10    0000.0000.0000.0000.0000  0          Gi6:10
000c.29b0.3e16 10    10    0000.0000.0000.0000.0000  0          10.7.7.7
000c.29ee.ec0d 10    10    0000.0000.0000.0000.0000  0          10.3.3.3

PE1# show l2vpn evpn mac detail
```

```
MAC Address:               000c.2993.130e
EVPN Instance:             10
Bridge Domain:             10
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop(s):               GigabitEthernet7 service instance 10
Label:                     22
Sequence Number:           0
MAC only present:          Yes
MAC Duplication Detection: Timer not running

MAC Address:               000c.29ee.ec0d
EVPN Instance:             10
Bridge Domain:             10
Ethernet Segment:          0000.0000.0000.0000.0000
Ethernet Tag ID:           0
Next Hop(s):               10.3.3.3
Local Address:             10.1.1.1
Label:                     19
Sequence Number:           0
MAC only present:          Yes
MAC Duplication Detection: Timer not running
```

Note In the above output, the next hop address of the remote MAC is the address of the provider edge device, if it is learnt remotely or the local interface if MAC address is learnt locally.

Use the following command to verify that Layer 2 Routing Information Base (RIB) has the required the MAC info:

```
PE1# show l2vpn l2route evpn mac
  EVI       ETag  Prod    Mac Address                    Next Hop(s) Seq Number
----- ---------- ----- -------------- ---------------------------- ----------
   10          0 L2VPN 000C.2993.130E                       Gi7:10          0
   10          0 L2VPN 000C.29AF.F904                       Gi6:10          0
   10          0   BGP 000C.29B0.3E16             L:19 IP:10.7.7.7          0
   10          0   BGP 000C.29EE.EC0D             L:19 IP:10.3.3.3          0
```

Note Remote MACs are learnt through BGP. In the above command output, the producer is BGP and local MACs are learned through Layer 2 VPN.

Use the following command to verify that Layer 2 FIB has received the MAC information from Layer 2 RIB, and bridge-domain and MFI are configured.

```
PE1# show l2fib bridge-domain 10 detail
Bridge Domain : 10
  Reference Count : 18
  Replication ports count : 4
  Unicast Address table size : 4
  IP Multicast Prefix table size : 4

  Flood List Information :
    Olist: Id 9225, Port Count 4

  Port Information :
```

```
    Serv Inst: Gi6:10
    Serv Inst: Gi7:10
    EVPN MPLS Encap: pathlist 107
    EVPN MPLS Encap: pathlist 101

  Unicast Address table information :
    Mac: 000c.2993.130e, Adjacency: Serv Inst: Gi7:10
    Mac: 000c.29af.f904, Adjacency: Serv Inst: Gi6:10
    Mac: 000c.29b0.3e16, Adjacency: EVPN MPLS Encap: pathlist 98
    Mac: 000c.29ee.ec0d, Adjacency: EVPN MPLS Encap: pathlist 104

  IP Multicast Prefix table information :
    Source: *, Group: 224.0.0.0/4, IIF:  , Adjacency: Olist: 9226, Ports: 0
    Source: *, Group: 224.0.0.0/24, IIF:  , Adjacency: Olist: 9225, Ports: 4
    Source: *, Group: 224.0.1.39, IIF:  , Adjacency: Olist: 9225, Ports: 4
    Source: *, Group: 224.0.1.40, IIF:  , Adjacency: Olist: 9225, Ports:
```

Use the following command to verify that the information on BGP route type 3 is sent to L2RIB:

```
PE1# show l2vpn l2route evpn imet
  EVI       ETAG  Prod  Router IP Addr  Type   Label       Tunnel ID
----- ---------- ----- --------------- ----- ------- ---------------
   10          0   BGP         10.3.3.3     6      18         10.3.3.3
   10          0   BGP         10.7.7.7     6      18         10.7.7.7
   10          0 L2VPN         10.1.1.1     6      23         10.1.1.1
```

Use the following command to verify MPLS forwarding:

```
PE1#show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
18         No Label   evpn(mc:bd 30)   305042        none       point2point
19         No Label   evpn(uc:bd 30)   7684          none       point2point
20         No Label   evpn(mc:bd 20)   542588        none       point2point
21         No Label   evpn(uc:bd 20)   13786         none       point2point
22         No Label   evpn(uc:bd 10)   6638          none       point2point
23         No Label   evpn(mc:bd 10)   277740        none       point2point
24         Pop Label  192.0.2.2-A      0             Gi1        192.0.2.2
25         Pop Label  192.0.2.2-A      0             Gi1        192.0.2.2
16001      16001      10.3.3.3/32       0             Gi1         192.0.2.2
16002      Pop Label  10.2.2.2/32       0             Gi1         192.0.2.2
16004      16004      10.7.7.7/32       0             Gi1         192.0.2.2

PE1# show ip bgp l2vpn evpn route-type 2
BGP routing table entry for [2][10.1.1.1:10][0][48][000C2993130E][0][*]/20, version 43
Paths: (1 available, best #1, table evi_10)
  Advertised to update-groups:
     2
  Refresh Epoch 1
  Local
    :: (via default) from 0.0.0.0 (10.1.1.1)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      EVPN ESI: 00000000000000000000, Label1 22
      Extended Community: RT:100:10
      rx pathid: 0, tx pathid: 0x0
BGP routing table entry for [2][10.1.1.1:10][0][48][000C29B03E16][0][*]/20, version 116
Paths: (1 available, best #1, table evi_10)
  Not advertised to any peer
  Refresh Epoch 3
  Local, (received & used), imported path from [2][10.7.7.7:10][0][48][000C29B03E16][0][*]/20
```

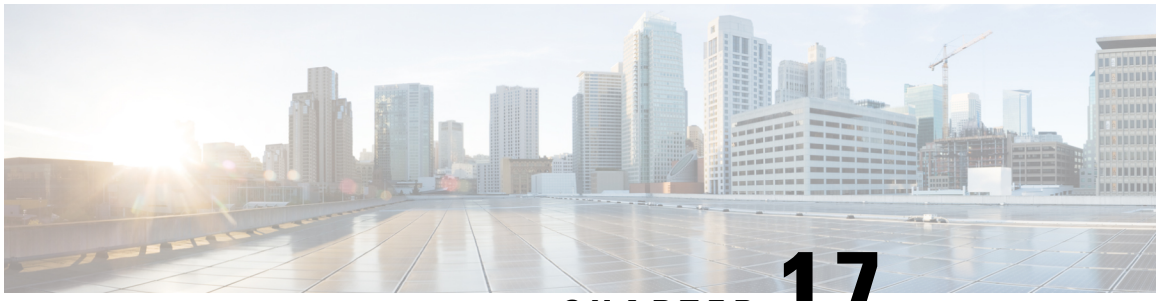
```
 (global)
    10.7.7.7 (metric 30) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 19
      Extended Community: RT:100:10
      Originator: 10.7.7.7, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0
BGP routing table entry for [2][10.1.1.1:10][0][48][000C29B03E16][0][*]/20, version 116
Paths: (1 available, best #1, table evi_10)
  Not advertised to any peer
  Refresh Epoch 3
  Local, (received & used), imported path from [2][10.7.7.7:10][0][48][000C29B03E16][0][*]/20
 (global)
    10.7.7.7 (metric 30) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 19
      Extended Community: RT:100:10
      Originator: 10.7.7.7, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0
BGP routing table entry for [2][10.1.1.1:10][0][48][000C29EEEC0D][0][*]/20, version 134
Paths: (1 available, best #1, table evi_10)
  Not advertised to any peer
  Refresh Epoch 3
  Local, (received & used), imported path from [2][10.3.3.3:10][0][48][000C29EEEC0D][0][*]/20
 (global)
    10.3.3.3 (metric 30) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      EVPN ESI: 00000000000000000000, Label1 19
      Extended Community: RT:100:10
      Originator: 10.3.3.3, Cluster list: 10.2.2.2
      rx pathid: 0, tx pathid: 0x0


PE1# show ip bgp l2vpn evpn route-type 3
BGP routing table entry for [3][10.1.1.1:10][0][32][10.1.1.1]/17, version 41
Paths: (1 available, best #1, table evi_10)
  Advertised to update-groups:
     2
  Refresh Epoch 1
  Local
    :: (via default) from 0.0.0.0 (10.1.1.1)
      Origin incomplete, localpref 100, weight 32768, valid, sourced, local, best
      Extended Community: RT:100:10
      PMSI Attribute: for EVPN, Flags: 0x0, Tunnel type: 6, length 4, label: 23 (vni 368)
tunnel parameters: 0101 0101
      rx pathid: 0, tx pathid: 0x0
BGP routing table entry for [3][10.1.1.1:10][0][32][10.3.3.3]/17, version 137
Paths: (1 available, best #1, table evi_10)
  Not advertised to any peer
  Refresh Epoch 3
  Local, (received & used), imported path from [3][10.3.3.3:10][0][32][10.3.3.3]/17 (global)

    10.3.3.3 (metric 30) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:10
      Originator: 10.3.3.3, Cluster list: 10.2.2.2
      PMSI Attribute: for EVPN, Flags: 0x0, Tunnel type: 6, length 4, label: 18 (vni 288)
tunnel parameters: 0303 0303
      rx pathid: 0, tx pathid: 0x0
BGP routing table entry for [3][10.1.1.1:10][0][32][10.7.7.7]/17, version 122
Paths: (1 available, best #1, table evi_10)
  Not advertised to any peer
  Refresh Epoch 3
  Local, (received & used), imported path from [3][10.7.7.7:10][0][32][10.7.7.7]/17 (global)
```

```
    10.7.7.7 (metric 30) (via default) from 10.2.2.2 (10.2.2.2)
      Origin incomplete, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:100:10
      Originator: 10.7.7.7, Cluster list: 10.2.2.2
      PMSI Attribute: for EVPN, Flags: 0x0, Tunnel type: 6, length 4, label: 18 (vni 288)
tunnel parameters: 0707 0707
      rx pathid: 0, tx pathid: 0x0
```

Additional References for EVPN Single-Homing

Standards and RFCs

Standard	Title
RFC 7432	BGP MPLS-Based Ethernet VPN



CHAPTER 17

Pseudowire Stitching

Pseudowire stitching is a technique where a pair of independent pseudowires are configured in such a way that they behave like a single point to point pseudowire. It is also called as multi-segment pseudowire (MS-PW).

Pseudowire stitching can be achieved using cross-connect.

Benefits of Pseudowire Stitching

Pseudowire stitching is useful in scenarios where a large network needs to be divided into small pieces, for example, core and metro side, each part of the network will be stitched to achieve end-to-end seamless connectivity.

Restrictions for Pseudowire Stitching

For Cisco ASR 900 RSP3 module, on pseudowire stitching point regular hardware programming is be seen because in this case pseudowire has to swap the label.

Configuring Pseudowire Stitching

Below is an example with three nodes connected:

Router IDs are:

- R1 - 10.1.1.1
- R2 - 2.2.2.2
- R3 - 3.3.3.3

Configuration on R1 node:

```
interface GigabitEthernet0/1/0
 no ip address
```

```
 negotiation auto
 service instance 1 ethernet
  encapsulation dot1q 1
  xconnect 2.2.2.2 100 encapsulation mpls
 !
```

Configuration on R2 node: (Stitching point)

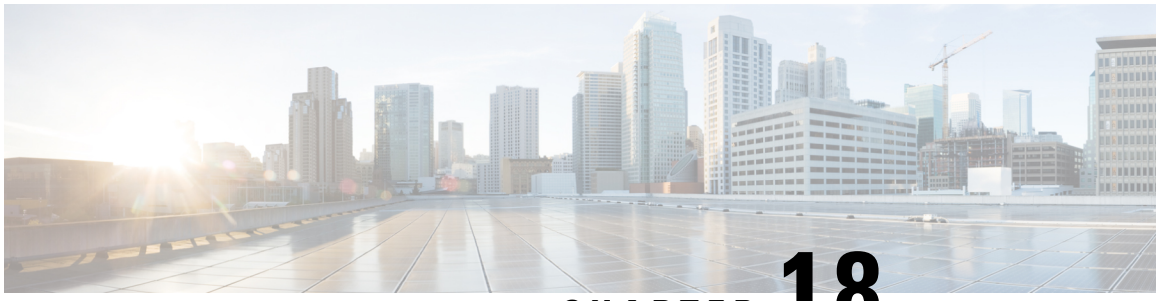
```
l2vpn xconnect context PW
 member 10.1.1.1 100 encapsulation mpls
 member 3.3.3.3 100 encapsulation mpls
```

Configuration on R3 node:

```
interface GigabitEthernet0/1/0
 no ip address
 negotiation auto
 service instance 1 ethernet
  encapsulation dot1q 1
  xconnect 2.2.2.2 100 encapsulation mpls
 !
```

Verifying Pseudowire Stitching

```
R2#show mpls l2transport vc

Local intf     Local circuit              Dest address    VC ID      Status
-------------  -------------------------- --------------- ---------- ----------
pw100010       3.3.3.3 100                10.1.1.1         100        UP
pw100009       10.1.1.1 100                3.3.3.3         100        UP
```



CHAPTER 18

On-Change Notifications for L2VPN Pseudowire

Table 15: Feature History

Feature Name	Release Information	Description
On-Change Notifications for L2VPN Pseudowire	Cisco IOS XE Bengaluru 17.5.1	This feature allows you to subscribe on-change Network Configuration Protocol (NETCONF) notifications for L2VPN pseudowire. You can generate an alert from a device when the pseudowire status changes.

Prior to Cisco IOS XE Bengaluru Release 17.5.1, it was not possible to externalize the internal IOS state for operational data. Thus, the on-change notifications were not generated for any change of state.

Starting with Cisco IOS XE Bengaluru Release 17.5.1, you can access the internal IOS state of the router to configure or view the running state of the router. The feature allows the externalization of the internal state of the router for the operational data. It helps in sending on-change notifications to the receiver for any chang of state, for example, when the pseudowire goes up or down. Thus, you can generate on-change NETCONF notifications for L2VPN pseudowire.

Use the Cisco-IOS-XE-l2vpn-pw-events operational module to configure the feature. The notification event and event data are included in the operational module.

IOS State

The IOS state can be divided into the following conceptual groups:

- Configuration or configuration state includes:
  - Feature default configuration state (use **show running-config all** command with no user configuration)
  - Non-default configuration state (use **show running-config** command)

- Nonvisible non-persistent feature configuration state (subscriber profiles)

- Operational state includes:
  - Feature state that is not configuration state (pseudowire status)

Telemetry and L2VPN Pseudowire

Telemetry is the process of measuring the state of the components in a system and transmitting it to a remote location for further processing and analysis. Event-driven Telemetry (EDT) optimizes data collected at the receiver by streaming data only when a state transition occurs (for example, stream data only when an interface state transitions, IP route updates, and so on).

For L2VPN pseudowire, the on-change notifications are sent when the pseudowire state changes with link flaps, configuration delete or add, and so on. For more information on the Telemetry feature, see the Programmability Configuration Guide, Cisco IOS XE Bengaluru 17.4.x.

The following events are supported:

- L2VPN Pseudowire status is up
- L2VPN Pseudowire status is down
- VPLS status is up
- VPLS status is down

Configuration Examples: On-Change Notifications for L2VPN Pseudowire

The following examples show the configurations for on-change notifications for L2VPN

On Router PE1:

Configuration for L2VPN Xconnect:

```
interface pseudowire1
encapsulation mpls
neighbor 2.2.2.2 1
!
interface pseudowire2
encapsulation mpls
neighbor 3.3.3.3 2
!
interface gi0/2/0
service instance 10 ethernet
encapsulation dot1q 10
rewrite ingress tag pop 1 symmetric
!
```

```
l2vpn xconnect context test
member pseudowire1 group 1 priority 1
member pseudowire2 group 2 priority 2
member gi0/2/0 service-instance 10
```

Sample Telemetry Configuration for Notification:

```
telemetry ietf subscription 1
encoding encode-tdl
filter tdl-uri /services;serviceName=iosevent/q_pw_session_state
stream native
update-policy on-change
receiver ip address x.x.x.x 45000 protocol native
```

Verification of On-Change Notifications for L2VPN Pseudowire Configuration

Establish Subscription via NETCONF for On-Change Notifications:

The following output displays the subscription creation and establishment that captures pseudowire events via NETCONF:

```
<establish-subscription
xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push"
xmlns:cyp="urn:cisco:params:xml:ns:yang:cisco-xe-ietf-yang-push-ext">
<stream>cyp:yang-notif-native</stream>
<yp:xpath-filter>/l2vpn-pw-ios-xe-events:l2vpn-pw-vc-status</yp:xpath-filter>
<yp:dampening-period>0</yp:dampening-period>
</establish-subscription>
```

Confirm Subscription Establishment:

This following output confirms subscription requested for events via NETCONF. This captures the pseudowire events when there is any change in the pseudowire state.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
<subscription-result xmlns="urn:ietf:params:xml:ns:yang:ietf-event- notifications"
xmlns:notif-bis="urn:ietf:params:xml:ns:yang:ietf-event-notifications">notif-bis:ok</subscription-result>
<subscription-id
xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications">2147483670</subscription-id>
</rpc-reply>
```

The output below shows the verification of on-change notifications for L2VPN psuedowire state configuration.

The following output shows when pseudowire goes Down:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
<eventTime>2020-08-25T07:32:35.52Z</eventTime>
<l2vpn-pw-vc-status xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-l2vpn-pw-events">
<vc-status>pw-vc-down</vc-status>
<vc-id>1022</vc-id>
<peer-ip>1.1.1.2</peer-ip>
</l2vpn-pw-vc-status>
</notification>
```

The following output shows when pseudowire comes Up:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
```

```
<eventTime>2020-08-25T07:55:35.52Z</eventTime>
<l2vpn-pw-vc-status xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-l2vpn-pw-events">
<vc-status>pw-vc-up</vc-status>
<vc-id>1022</vc-id>
<peer-ip>1.1.1.2</peer-ip>
</l2vpn-pw-vc-status>
</notification>
```