Implementing Virtual Private LAN Services

This module provides the conceptual and configuration information for Virtual Private LAN Services (VPLS) on Cisco IOS XR software. VPLS supports Layer 2 VPN technology and provides transparent multipoint Layer 2 connectivity for customers.

This approach enables service providers to host a multitude of new services such as broadcast TV, Layer 2 VPNs.

For MPLS Layer 2 virtual private networks (VPNs), see Implementing MPLS Layer 2 VPNs module.

Note For more information about MPLS Layer 2 VPN on Cisco IOS XR software and for descriptions of the commands listed in this module, see the "Related Documents" section. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing Virtual Private LAN Services on Cisco IOS XR Configuration Module

Release	Modification
Release 3.7.0	This feature was introduced.
Release 3.8.0	Support for the bridging funtionality feature (VPLS based) and pseudowire redundancy, was added on the Cisco CRS-1 router.
Release 3.9.0	The following features were added: • Blocking unknown unicast flooding. • Disabling MAC flush.
Release 4.1.1	Support for these features was added: • Mutlisegment Pseudowire • Pseudowire Redundancy • Pseudowire Headend
Release 4.2.1	The pseudowire headend (PWHE) feature was enhanced to support: • eBGP on PWHE interfaces • IPv6 Unicast on PWHE
Release 4.3.0	Support was added for these features: • Flow Aware Transport (FAT) Pseudowire • Pseudowire Grouping

Contents

Prerequisites for Implementing Virtual Private LAN Services

Before you configure VPLS, ensure that the network is configured as follows:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

  If you need assistance with your task group assignment, contact your system administrator.
- Configure IP routing in the core so that the provider edge (PE) routers can reach each other through IP.
- Configure MPLS and Label Distribution Protocol (LDP) in the core so that a label switched path (LSP) exists between the PE routers.
- Configure a loopback interface to originate and terminate Layer 2 traffic. Make sure that the PE routers can access the other router's loopback interface.

Note The loopback interface is not needed in all cases. For example, tunnel selection does not need a loopback interface when VPLS is directly mapped to a TE tunnel.

Restrictions for Implementing Virtual Private LAN Services

The following restrictions are listed for implementing VPLS:

- All attachment circuits in a bridge domain on an Engine 3 line card must be the same type (for example, port, dot1q, qinq, or qinany), value (VLAN ID), and EtherType (for example, 0x8100, 0x9100, or 0x9200).
- The Engine 3 line cards, cannot simultaneously have attachment circuits and MPLS-enabled on any one of its interfaces. The line card cannot be Edge-facing and Core-facing at the same time.
- The line card requires ternary content addressable memory (TCAM) Carving configuration.
- Virtual Forwarding Instance (VFI) names have to be unique, because a bridge domain can have only one VFI.
- A PW cannot belong to both a peer-to-peer (P2P) cross-connect group and a VPLS bridge-domain. This means that the neighboring IP address and the pseudowire ID have to be unique on the router, because the pseudowire ID is signaled to the remote provider edge.

- You cannot manually set up a PW on one PE and use auto-discovery on the other PE to configure the same PW in the other direction.

For the Engine 5 line card, version 1 of the Ethernet SPA does not support QinQ mode and QinAny mode.

Note For the Engine 5 line card, version 2 of the Ethernet SPA supports all VLAN modes, such as VLAN mode, QinQ mode, or QinAny mode.

Information About Implementing Virtual Private LAN Services

To implement Virtual Private LAN Services (VPLS), you should understand the following concepts:

Virtual Private LAN Services Overview

Virtual Private LAN Service (VPLS) enables geographically separated local-area network (LAN) segments to be interconnected as a single bridged domain over an MPLS network. The full functions of the traditional LAN such as MAC address learning, aging, and switching are emulated across all the remotely connected LAN segments that are part of a single bridged domain. A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

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

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

After provisioning attachment circuits, neighbor relationships across the MPLS network for this specific instance are established through a set of manual commands identifying the end PEs. When the neighbor association is complete, a full mesh of pseudowires is established among the network-facing provider edge devices, which is a gateway between the MPLS core and the customer domain.

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

VPLS requires the provider edge device to be MPLS-capable. The VPLS provider edge device holds all the VPLS forwarding MAC tables and Bridge Domain information. In addition, it is responsible for all flooding broadcast frames and multicast replications.

Note VPLS with Traffic Engineering Fast Reroute (TE FRR) is not supported.

VPLS for an MPLS-based Provider Core

VPLS is a multipoint Layer 2 VPN technology that connects two or more customer devices using bridging techniques. The VPLS architecture allows for the end-to-end connection between the Provider Edge (PE) routers to provide Multipoint Ethernet Services.

VPLS requires the creation of a bridge domain (Layer 2 broadcast domain) on each of the PE routers. The access connections to the bridge domain on a PE router are called *attachment circuits* (AC).

The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

The MPLS/IP provider core simulates a virtual bridge that connects the multiple attachment circuits on each of the PE devices together to form a single broadcast domain. A VFI is created on the PE router for each VPLS instance. The PE routers make packet-forwarding decisions by looking up the VFI of a particular VPLS instance. The VFI acts like a virtual bridge for a given VPLS instance. More than one attachment circuit belonging to a given VPLS are connected to the VFI. The PE router establishes emulated VCs to all the other PE routers in that VPLS instance and attaches these emulated VCs to the VFI. Packet forwarding decisions are based on the data structures maintained in the VFI.

Hierarchical VPLS

Hierarchical VPLS (H-VPLS) is an extension of basic VPLS that provides scaling and operational benefits. H-VPLS provides a solution to deliver Ethernet multipoint services over MPLS. H-VPLS partitions a network into several edge domains that are interconnected using an MPLS core. The use of Ethernet switches at the edge offers significant technical and economic advantages. H-VPLS also allows Ethernet point-to-point and multipoint Layer 2 VPN services, as well as Ethernet access to high-speed Internet and IP VPN services.

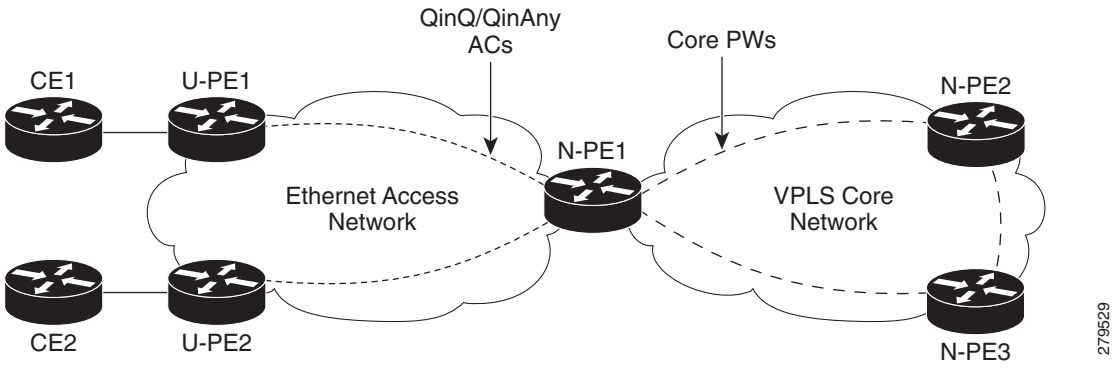
Two flavors of H-VPLS are:

- Ethernet access in the edge domain
- MPLS access in the edge domain

H-VPLS with Ethernet Access QinQ or QinAny

Figure 1 shows Ethernet access for H-VPLS. The edge domain can be built using Ethernet switches and techniques such as QinQ. Using Ethernet as the edge technology simplifies the operation of the edge domain and reduces the cost of the edge devices.

Figure 13 Ethernet Access for H-VPLS

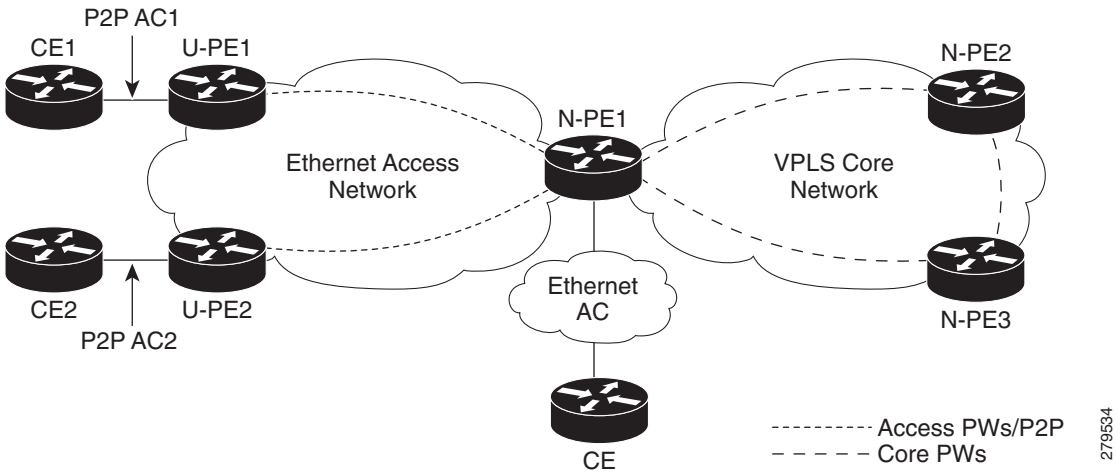


H-VPLS with PW-access

Figure 14 shows pseudowire (PW) access for H-VPLS. The edge domain can be an MPLS access network. In this scenario, the U-PE device carries the customer traffic from attachment circuits (AC) over the point to point (p2p) pseudowires. The p2p pseudowires terminate in a bridge domain configured on the N-PE device.

Access PW is configured as a member directly under a bridge domain. A bridge-domain in N-PE1 can have multiple ACs (physical/VLAN Ethernet ports), multiple access PWs and one VFI (consisting of core PWs) as members, is depicted in Figure 14.

Figure 14 PW access for H-VPLS



Signaling

An important aspect of VPN technologies, including VPLS, is the ability of network devices to automatically signal to other devices about an association with a particular VPN, often referred to as *signaling mechanisms*.

The implementation of VPLS in a network requires the establishment of a full mesh of pseudowires between the provider edge (PE) routers. The signaling of pseudowires between provider edge devices, described in *draft-ietf-l2vpn-vpls-ldp-09*, uses targeted LDP sessions to exchange label values and attributes and to setup the pseudowires. LDP is an efficient mechanism for signaling pseudowire status for Ethernet point-to-point and multipoint services.

Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling

The Cisco IOS Software encodes the NLRI length in the fist byte in bits format in the BGP Update message. However, the Cisco IOS XR Software interprets the NLRI length in 2 bytes. Therefore, when the BGP neighbor with VPLS-VPWS address family is configured between the IOS and the IOS XR, NLRI mismatch can happen, leading to flapping between neighbors. To avoid this conflict, IOS supports **prefix-length-size 2** command that needs to be enabled for IOS to work with IOS XR. When the **prefix-length-size 2** command is configured in IOS, the NLRI length is encoded in bytes. This configuration is mandatory for IOS to work with IOS XR.

This is a sample IOS configuration with the **prefix-length-size 2** command:

```
router bgp 1
 address-family l2vpn vpls
  neighbor 5.5.5.2 activate
  neighbor 5.5.5.2 prefix-length-size 2 --------> NLRI length = 2 bytes
 exit-address-family
```

Bridge Domain

The native bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports (including VFI). Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain. A learned address is aged out. Incoming frames are mapped to a bridge domain, based on either the ingress port or a combination of both an ingress port and a MAC header field.

By default, split horizon is enabled on a bridge domain. In other words, any packets that are coming on either the attachment circuits or pseudowires are not returned on the same attachment circuits or pseudowires. In addition, the packets that are received on one pseudowire are not replicated on other pseudowires in the same VFI.

MAC Address-related Parameters

The MAC address table contains a list of the known MAC addresses and their forwarding information. In the current VPLS design, the MAC address table and its management are distributed. In other words, a copy of the MAC address table is maintained on the route processor (RP) card and the line cards.

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

- MAC Address Flooding, page VPC-93

MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To obtain flooding within VPLS broadcast models, all unknown unicast, broadcast, and multicast frames are flooded over the corresponding pseudowires and to all attachment circuits. Therefore, a PE must replicate packets across both attachment circuits and pseudowires.

MAC Address-based Forwarding

To forward a frame, a PE must associate a destination MAC address with a pseudowire or attachment circuit. This type of association is provided through a static configuration on each PE or through dynamic learning, which is flooded to all bridge ports.

Note In this case, split horizon forwarding applies; for example, frames that are coming in on an attachment circuit or pseudowire are not sent out of the same attachment circuit or pseudowire. The pseudowire frames, which are received on one pseudowire, are replicated on to other attachment circuits, VFI pseudowires and access pseudowires.

MAC Address Source-based Learning

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

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. The updated MAC tables are sent to all line cards (LCs) and program the hardware for the router.

The number of learned MAC addresses is limited through configurable per-port and per-bridge domain MAC address limits.

MAC Address Aging

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

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

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

MAC Address Limit

The MAC address limit is used to limit the number of learned MAC addresses. The limit is set at the bridge domain level and the port level. When the MAC address limit is violated, the system is configured to take one of the actions that are listed in Table 3.

Table 3 MAC Address Limit Actions

Action	Description
Limit flood	Discards the new MAC addresses.
Limit no-flood	Discards the new MAC addresses. Flooding of unknown unicast packets is disabled.
Shutdown	Disables the bridge domain or bridge port. When the bridge domain is down, none of the bridging functions, such as learning, flooding, forwarding, and so forth take place for the bridge domain. If a bridge port is down as a result of the action, the interface or pseudowire representing the bridge port remains up but the bridge port is not participating in the bridge. When disabled, the port or bridge domain is manually brought up by using an EXEC CLI.

When a limit is exceeded, the system is configured to perform the following notifications:

- Syslog (default)
- Simple Network Management Protocol (SNMP) trap
- Syslog and SNMP trap
- None (no notification)

To clear the MAC limit condition, the number of MACs must go below 75 percent of the configured limit.

MAC Address Withdrawal

For faster VPLS convergence, you can remove or unlearn the MAC addresses that are learned dynamically. The Label Distribution Protocol (LDP) Address Withdrawal message is sent with the list of MAC addresses, which need to be withdrawn to all other PEs that are participating in the corresponding VPLS service.

For the Cisco IOS XR VPLS implementation, a portion of the dynamically learned MAC addresses are cleared by using the MAC addresses aging mechanism by default. The MAC address withdrawal feature is added through the LDP Address Withdrawal message. To enable the MAC address withdrawal feature,

use the **withdrawal** command in l2vpn bridge group bridge domain MAC configuration mode. To verify that the MAC address withdrawal is enabled, use the **show l2vpn bridge-domain** command with the **detail** keyword.

Note By default, the LDP MAC Withdrawal feature is enabled on Cisco IOS XR.

The LDP MAC Withdrawal feature is generated due to the following events:

- Attachment circuit goes down. You can remove or add the attachment circuit through the CLI.
- MAC withdrawal messages are received over a VFI pseudowire and are not propagated over access pseudowires. RFC 4762 specifies that both wildcards (by means of an empty Type, Length and Value [TLV]) and a specific MAC address withdrawal. Cisco IOS XR software supports only a wildcard MAC address withdrawal.

LSP Ping over VPWS and VPLS

For Cisco IOS XR software, the existing support for the Label Switched Path (LSP) ping and traceroute verification mechanisms for point-to-point pseudowires (signaled using LDP FEC128) is extended to cover the pseudowires that are associated with the VFI (VPLS). Currently, the support for the LSP ping and traceroute is limited to manually configured VPLS and access pseudowires (signaled using LDP FEC128). Virtual Circuit Connection Verification (VCCV) is also supported on access pseudowires. For information about VCCV support and the **ping mpls pseudowire** command, see *Cisco IOS XR MPLS Command Reference for the Cisco XR 12000 Series Router.*

Pseudowire Redundancy for P2P AToM Cross-Connects

Backup pseudowires (PW) are associated with the corresponding primary pseudowires. A backup PW is not programmed to forward data when inactive. It is activated only if a primary PW fails. This is known as *pseudowire redundancy.* The primary reason for backing up a PW is to reduce traffic loss when a primary PW fails. When the primary PW is active again, it resumes its activity.

A primary PW can be associated with only one backup PW. Similarly, a backup PW can be associated with only one primary PW.

Note This feature is supported only for an AToM instance on the Cisco XR 12000 Series Router, and for an EoMPLS instance on the Cisco CRS-1 router.

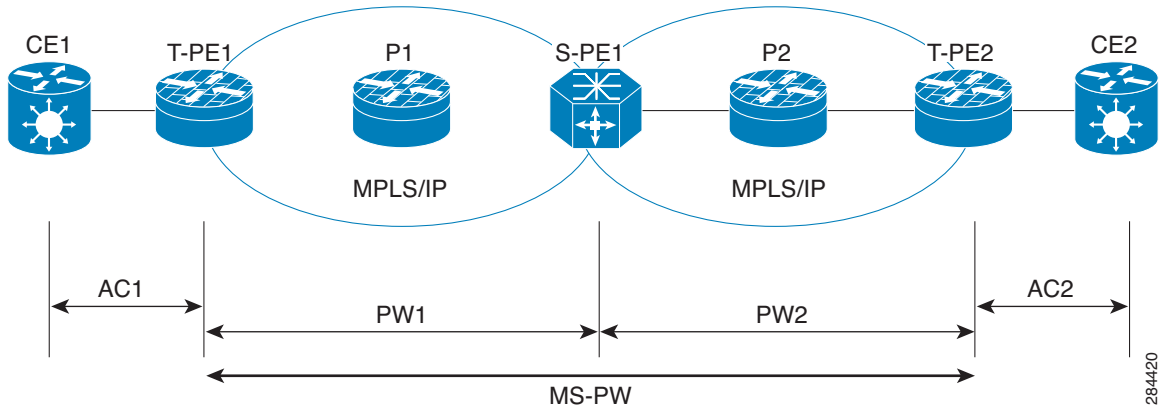
Multisegment Pseudowire

Pseudowires transport Layer 2 protocol data units (PDUs) across a public switched network (PSN). A multisegment pseudowire is a static or dynamically configured set of two or more contiguous pseudowire segments. These segments act as a single pseudowire, allowing you to:

- Manage the end-to-end service by separating administrative or provisioning domains.
- Keep IP addresses of provider edge (PE) nodes private across interautonomous system (inter-AS) boundaries. Use IP address of autonomous system boundary routers (ASBRs) and treat them as pseudowire aggregation routers. The ASBRs join the pseudowires of the two domains.

A multisegment pseudowire can span either an inter-AS boundary or two multiprotocol label switching (MPLS) networks.

Figure 15 Multisegment Pseudowire: Example



A pseudowire is a tunnel between two PE nodes. There are two types of PE nodes:

- A Switching PE (S-PE) node
  - Terminates PSN tunnels of the preceding and succeeding pseudowire segments in a multisegment pseudowire.
  - Switches control and data planes of the preceding and succeeding pseudowire segments of the multisegment pseudowire.
- A Terminating PE (T-PE) node
  - Located at both the first and last segments of a multisegment pseudowire.
  - Where customer-facing attachment circuits (ACs) are bound to a pseudowire forwarder.

Note Every end of a multisegment pseudowire must terminate at a T-PE.

A multisegment pseudowire is used in two general cases when:

- It is not possible to establish a PW control channel between the source and destination PE nodes.

  For the PW control channel to be established, the remote PE node must be accessible. Sometimes, the local PE node may not be able to access the remote node due to topology, operational, or security constraints.
  A multisegment pseudowire dynamically builds two discrete pseudowire segments and performs a pseudowire switching to establish a PW control channel between the source and destination PE nodes.
- Pseudowire Edge To Edge Emulation (PWE3) signaling and encapsulation protocols are different.

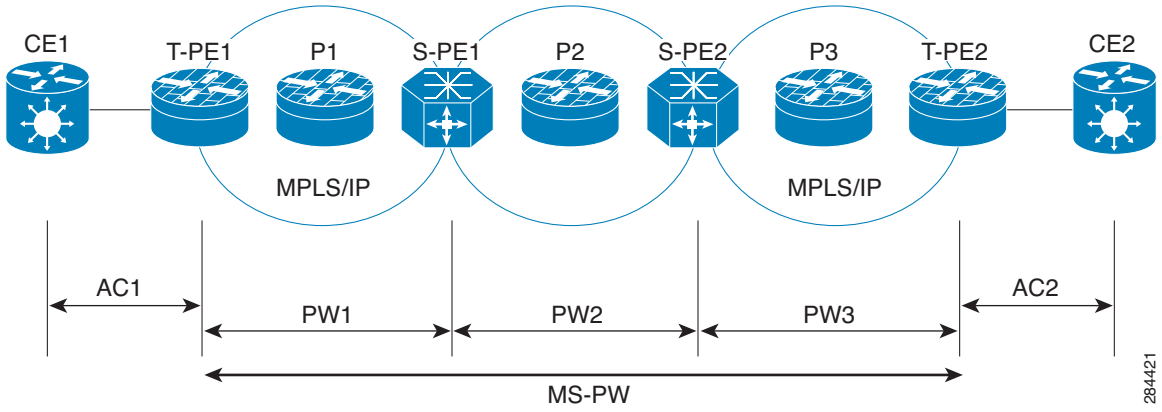
  The PE nodes are connected to networks employing different PW signaling and encapsulation protocols. Sometimes, it is not possible to use a single segment PW.
  A multisegment pseudowire, with the appropriate interworking performed at the PW switching points, enables PW connectivity between the PE nodes in the network.

Pseudowire Switching

Pseudowire Switching connects two or more contiguous pseudowire segments to form an end-to-end multihop pseudowire. This end-to-end pseudowire functions as a single point-to-point pseudowire. It allows you to extend pseudowires across an inter-AS boundary or across two separate networks.

The edge to edge PW may traverse several switching points, in separate administrative domains. For management and troubleshooting reasons you can record information about the switching points that the PW traverses. This is accomplished by using a PW switching point TLV.

Figure 16 Pseudowire Switching: Example



In the above figure, the multisegment pseudowire is established between T-PE1 and T-PE2 with S-PE1 and S-PE2 as switching points. The pw-id 1 is between T-PE1 and S-PE1, pw-id 2 is between S-PE1 and S-PE2 and pw-id 3 is between S-PE2 and T-PE2.

Consider a packet traversal from T-PE1 to T-PE2:

1. T-PE1 sends label mapping message without a PW Switching Point TLV signal.
2. S-PE1 adds a PW Switching Point TLV signal with 4 sub-TLVs:
   a. description string
   b. pw-id 1
   c. S-PE1 IP address, which is the local address. This is the local router-id of the S-PE.
   d. T-PE1 IP address, which is the remote address
3. S-PE2 adds a PW Switching Point TLV signal with 3 sub-TLVs:
   a. description string
   b. pw-id 2
   c. S-PE2 IP address, which is the local address.
   d. No remote address because S-PE1 address is already present in the message as local IP address in the last TLV.
4. T-PE2 gets the label mapping message with 2 PW Switching TLVs.

Sometimes, you do not expose the information about previous S-PEs to the next S-PE for security reasons. By default, an S-PE appends its information to the PW Switching Point TLV signal. When "hiding" option is enabled on a PW segment using the "switching tlv hide" command, an S-PE sends a label mapping message without any PW Switching Point TLVs.

Pseudowire Redundancy

Pseudowire redundancy allows you to configure your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. 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 connectivity between 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.

Pseudowire redundancy enables you to set up backup pseudowires. You can configure the network with redundant pseudowires and redundant network elements.

Prior to the failure of the primary pseudowire, the ability to switch traffic to the backup pseudowire is used to handle a planned pseudowire outage, such as router maintenance.

Note Pseudowire redundancy is provided only for point-to-point Virtual Private Wire Service (VPWS) pseudowires.

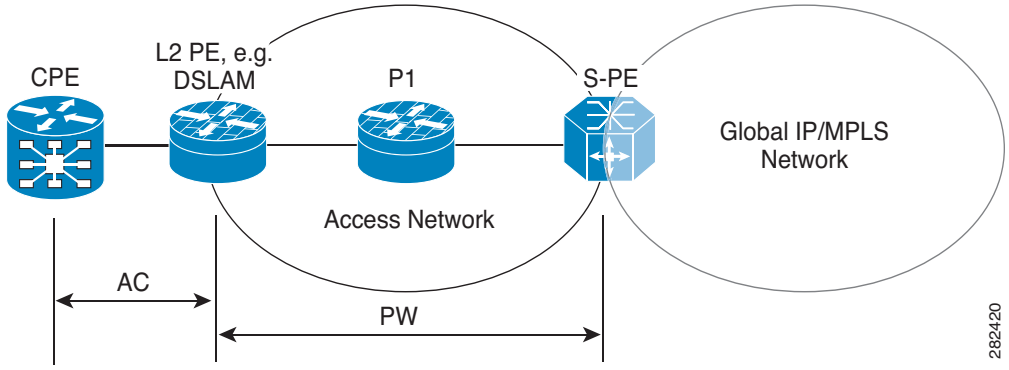
Pseudowire Headend

Pseudowires (PWs) enable payloads to be transparently carried across IP/MPLS packet-switched networks (PSNs). Service providers are now extending PW connectivity into the access and aggregation regions of their networks. PWs are regarded as simple and manageable lightweight tunnels for returning customer traffic into core networks.

The PW headend (PWHE) feature provides a Layer 3 (L3) virtual interface representation of a PW on an service provider edge (PE), that allows the backhaul of customer packets over PWs and the application of L3 features, such as QoS (for example: policing and shaping), and access lists (ACLs) on customer packets on the PW.

The PWHE virtual interface originates as a PW on an access node (the Layer 2 PW feeder node) and terminates on a Layer 3 service instance, such as a VRF instance, on the service provider router. At the service PE, IP traffic on the PW (from a remote customer PE via the access network) is forwarded onto the IP/MPLS backbone and traffic from the IP/MPLS backbone, is forwarded onto the PWHE L3 interface towards the customer PE (via the access network).

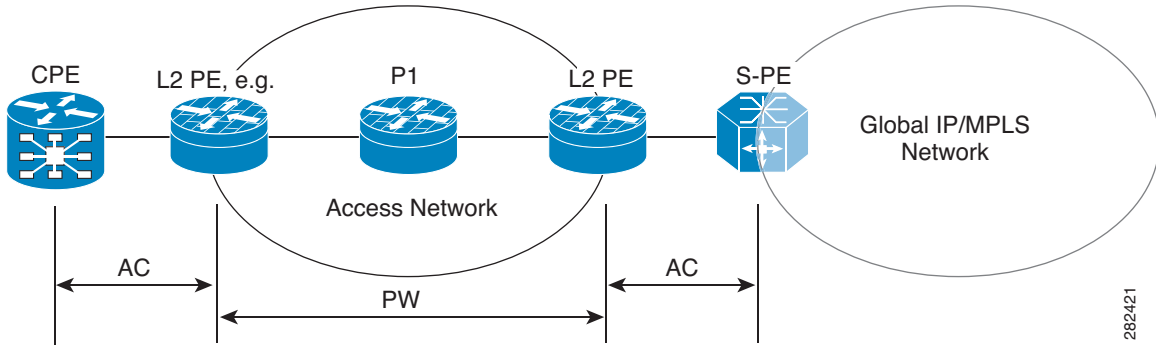
Figure 17 Example with PWHE



Note The PW is from L2 PE node to the Service PE (S-PE), but the L3 adjacency on each PWHE interface is configured between the service PE and the customer PE.

The PWHE feature allows you to replace a two node solution with a single node. Figure 18 illustrates a scenario wherein, without PWHE, an L2 PE node is required. The L2 PE node terminates the PW and connects to the service PE (from the L2 PE) via an attachment circuit (AC) that terminates as an L3 interface on the service PE.

Figure 18 Example without PWHE



PWHE Interfaces

The virtual circuit (VC) types supported for the PW are types 4, 5 and 11. The PWHE acts as broadcast interface with VC types 4 (VLAN tagged) and 5 (Ethernet port/Raw), whereas with VC type 11 (IP Interworking), the PWHE acts as a point-to-point interface.

eBGP Support on PWHE interfaces

To enable access CE to communicate with service PE, you need to configure eBGP on the PWHE interface. Enabling eBGP unblocks the path for all control packets (including eBGP) over PW-Ether (that is, for IPv4 and IPV6, and PW-IW interface for IPv4 only.)

IPv6 Unicast Support on PWHE

IPv6 Support is added by supporting the 6PE and 6VPE features. The Core network is either MPLS based or IP based. The IPv6 interfaces support both VC type 4 and 5.

Flow Aware Transport Pseudowire (FAT PW) Overview

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

Flow-Aware Transport Pseudowires (FAT PW) provide the capability to identify individual flows within a pseudowire and provide routers the ability to use these flows to loadbalance traffic. FAT PWs are used to loadbalance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created

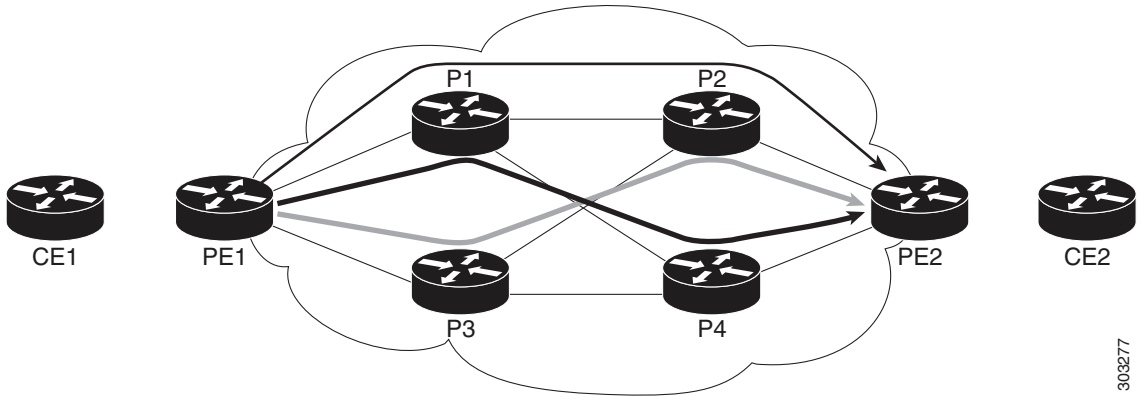
based on indivisible packet flows entering a pseudowire; and is inserted as the lower most label in the packet. Routers can use the flow label for loadbalancing which provides better traffic distribution across ECMP paths or link-bundled paths in the core.

An additional label is added to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from the source and destination IP addresses. 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). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

All core routers perform load balancing based on the flow-label in the FAT PW. Therefore, it is possible to distribute flows over ECMPs and link bundles.

Figure 19 shows a network with equal-cost multi-paths (ECMP).

Figure 19 Equal Cost Multi Path network



In Figure 19, traffic received from CE1 on PE1 load balances either P1 or P3, because the cost of links is equal. Further on P1, traffic flows from either P2 or P4. Similarly, P3 load balances either P2 or P4. The flow label helps to maximize load balancing on the P routers, throughout the network. The Ingress PE routers are responsible to add flow label whereas the Egress PE routers remove the flow label.

Pseudowire Grouping

When pseudowires (PW) are established, each PW is assigned a group ID that is common for all PWs created from the same physical port. Hence, when the physical port becomes non-functional or is deleted, L2VPN sends a single message to advertise the status change of all PWs belonging to the group. A single L2VPN signal thus avoids a lot of processing and loss in reactivity.

Note Pseudowire grouping is disabled by default.

How to Implement Virtual Private LAN Services

This section describes the tasks that are required to implement VPLS:

- Configuring a Bridge Domain, page VPC-101

Configuring a Bridge Domain

These topics describe how to configure a bridge domain:

Creating a Bridge Domain

Perform this task to create a bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Pseudowire

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **exit**
7. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi-name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures the virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.
		• Use the *vfi-name* argument to configure the name of the specified virtual forwarding interface.

	Command or Action	Purpose

Step 6	**exit**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# exit`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Exits the current configuration mode.

Step 7	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Backup Pseudowire

Perform this task to configure a backup pseudowire for a point-to-point neighbor.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group name*

4. **p2p** *xconnect name*
5. **neighbor** *ip-address* **pw-id** *number*
6. **backup neighbor** *ip-address* **pw-id** *number*
7. **end**

   or

   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**xconnect group** *group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group A`
	`RP/0/0/CPU0:router(config-l2vpn-xc)#`
		Enters the name of the cross-connect group.

Step 4	**p2p** *xconnect name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

Step 5	**neighbor** *ip-address* **pw-id** *number*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose

Step 6	**backup neighbor** *ip-address* **pw-id** *number*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# backup neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#`
		Configures the backup pseudowire for the point-to-point neighbor.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Backup Disable Delay

The Backup Disable Delay function specifies the time for which the primary pseudowire in active state waits before it takes over for the backup pseudowire. Perform this task to configure a disable delay.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class name*
4. **backup disable delay** *seconds*
5. **exit**
6. **xconnect group** *group name*
7. **p2p** *xconnect name*
8. **neighbor** *ip-address* **pw-id** *number*
9. **pw-class** *class name*

10. **backup neighbor** *ip-address* **pw-id** *number*
11. **end**

    or

    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**pw-class** *class_1*
	
	Example:
	`RP/0/RP0/CPU0:router(config-l2vpn)# pw-class class_1`
	`RP/0/RP0/CPU0:router(config-l2vpn-pwc)#`
		Configures the pseudowire class name.

Step 4	**backup disable delay** *seconds*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20`
	`RP/0/0/CPU0:router(config-l2vpn-pwc)#`
		Specifies how long a backup pseudowire virtual circuit (VC) should wait before resuming operation after the primary pseudowire VC becomes nonfunctional.

Step 5	**exit** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# exit`	Exits the pseudowire class submode.

Step 6	**xconnect group** *group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group A`
	`RP/0/0/CPU0:router(config-l2vpn-xc)#`
		Enters the name of the cross-connect group.

Step 7	**p2p** *xconnect name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

	Command or Action	Purpose

Step 8	**neighbor** *ip-address* **pw-id** *number*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire segment for the cross-connect.

Step 9	**pw-class** *class_1*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class class_1`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire class name.

Step 10	**backup neighbor** *ip-address* **pw-id** *number*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#`
		Configures the backup pseudowire for the point-to-point neighbor.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating Members with a Bridge Domain

After a bridge domain is created, perform this task to assign interfaces to the bridge domain. The following types of bridge ports are associated with a bridge domain:

- Ethernet and VLAN

- VFI

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **interface** *type interface-path-id*
6. **static-mac-address** {*MAC-address*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**interface** *type interface-path-id*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)#`
		Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

	Command or Action	Purpose
Step 6	**static-mac-address** {*MAC-address*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# static-mac-address 1.1.1`	Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Bridge Domain Parameters

To configure the bridge domain parameters, associate the following parameters with a bridge domain:

- Maximum transmission unit (MTU)—Specifies that all members of a bridge domain have the same MTU. The bridge domain member with a different MTU size is not used by the bridge domain even though it is still associated with a bridge domain.
- Flooding—Enables or disables flooding on the bridge domain. By default, flooding is enabled.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **flooding disable**

6. **mtu** *bytes*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**flooding disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)# flooding disable`	Configures flooding for traffic at the bridge domain level or at the bridge port level.

	Command or Action	Purpose
Step 6	**mtu** *bytes* **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mtu 1000`	Adjusts the maximum packet size or maximum transmission unit (MTU) size for the bridge domain. • Use the *bytes* argument to specify the MTU size, in bytes. The range is from 64 to 65535.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling a Bridge Domain

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **shutdown**
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

	Command or Action	Purpose
Step 5	**shutdown** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`	Shuts down a bridge domain to bring the bridge and all attachment circuits and pseudowires under it to admin down state.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Blocking Unknown Unicast Flooding

Perform this task to disable flooding of unknown unicast traffic at the bridge domain level.

You can disable flooding of unknown unicast traffic at the bridge domain, bridge port or access pseudowire levels. By default, unknown unicast traffic is flooded to all ports in the bridge domain.

Note If you disable flooding of unknown unicast traffic on the bridge domain, all ports within the bridge domain inherit this configuration. You can configure the bridge ports to override the bridge domain configuration.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group name*
4. **bridge-domain** *bridge-domain name*
5. **flooding unknown-unicast disable**
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

	Command or Action	Purpose
Step 5	**flooding unknown-unicast disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)# flooding unknown-unicast disable`	Disables flooding of unknown unicast traffic at the bridge domain level.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a Layer 2 Virtual Forwarding Instance

These topics describe how to configure a Layer 2 virtual forwarding instance (VFI):

- Adding the Virtual Forwarding Instance Under the Bridge Domain, page VPC-117
- Associating Pseudowires with the Virtual Forwarding Instance, page VPC-118
- Associating a Virtual Forwarding Instance to a Bridge Domain, page VPC-120
- Attaching Pseudowire Classes to Pseudowires, page VPC-122
- Configuring Any Transport over Multiprotocol Pseudowires By Using Static Labels, page VPC-124
- Disabling a Virtual Forwarding Instance, page VPC-126

Adding the Virtual Forwarding Instance Under the Bridge Domain

Perform this task to create a Layer 2 Virtual Forwarding Instance (VFI) on all provider edge devices under the bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

	Command or Action	Purpose

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-vpn)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-vpn)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating Pseudowires with the Virtual Forwarding Instance

After a VFI is created, perform this task to associate one or more pseudowires with the VFI.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** *A.B.C.D* {**pw-id** *value*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

	Command or Action	Purpose

Step 6	**neighbor** *A.B.C.D* {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Associating a Virtual Forwarding Instance to a Bridge Domain

Perform this task to associate a VFI to be a member of a bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}

7. **static-mac-address** {*MAC address*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**vfi** *vfi name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6	**neighbor** *A.B.C.D* {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

	Command or Action	Purpose
Step 7	**static-mac-address** {*MAC address*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# static-mac-address 1.1.1`	Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Attaching Pseudowire Classes to Pseudowires

Perform this task to attach a pseudowire class to a pseudowire.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
7. **pw-class** {*class name*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

	Command or Action	Purpose
Step 7	**pw-class** {*class name*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada`	Configures the pseudowire class template name to use for the pseudowire.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Any Transport over Multiprotocol Pseudowires By Using Static Labels

Perform this task to configure the Any Transport over Multiprotocol (AToM) pseudowires by using the static labels. A pseudowire becomes a static AToM pseudowire by setting the MPLS static labels to local and remote.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **neighbor** {*A.B.C.D*} {**pw-id** *value*}

7. **mpls static label** {**local** *value*} {**remote** *value*}
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#`
		Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).
		• Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
		• Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

	Command or Action	Purpose
Step 7	**mpls static label** {**local** *value*} {**remote** *value*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# mpls static label local 800 remote 500`	Configures the MPLS static labels and the static labels for the access pseudowire configuration. You can set the local and remote pseudowire labels.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Disabling a Virtual Forwarding Instance

Perform this task to disable a VFI. When a VFI is disabled, all the previously established pseudowires that are associated with the VFI are disconnected. LDP advertisements are sent to withdraw the MAC addresses that are associated with the VFI. However, you can still attach or detach attachment circuits with a VFI after a shutdown.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** {*vfi name*}
6. **shutdown**

7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**vfi** {*vfi name*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6	**shutdown** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown`	Disables the virtual forwarding interface (VFI).

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the state of the VFI. For example, if you shut down the VFI, the VFI is shown as shut down under the bridge domain.

Configuring the MAC Address-related Parameters

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

- Configuring the MAC Address Source-based Learning, page VPC-129
- Disabling the MAC Address Withdrawal, page VPC-131
- Configuring the MAC Address Limit, page VPC-133
- Configuring the MAC Address Aging, page VPC-135
- Disabling MAC Flush at the Bridge Port Level, page VPC-138

The MAC table attributes are set for the bridge domains.

Configuring the MAC Address Source-based Learning

Perform this task to configure the MAC address source-based learning.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **learning disable**
7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

	Command or Action	Purpose
Step 6	**learning disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# learning disable`	Overrides the MAC learning configuration of a parent bridge or sets the MAC learning configuration of a bridge.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details that the MAC address source-based learning is disabled on the bridge.

Disabling the MAC Address Withdrawal

Perform this task to disable the MAC address withdrawal for a specified bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **withdraw** { **access-pw disable** | **disable** }
7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

	Command or Action	Purpose
Step 6	**withdraw { access-pw disable | disable }** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# withdraw access-pw disable`	Disables the MAC address withdrawal for the specified bridge domain. **Note** Mac address withdrawal is generated when the access pseudowire is not operational.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8	**show l2vpn bridge-domain** [**detail**] **Example:** `P/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays detailed sample output to specify that the MAC address withdrawal is enabled. In addition, the sample output displays the number of MAC withdrawal messages that are sent over or received from the pseudowire.

The following sample output shows the MAC address withdrawal fields:

```
RP/0/0/CPU0:router# show l2vpn bridge-domain detail

Bridge group: siva_group, bridge-domain: siva_bd, id: 0, state: up, ShgId: 0, MSTi: 0
  MAC Learning: enabled
  MAC withdraw: enabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown Unicast: enabled
  MAC address aging time: 300 s Type: inactivity
  MAC address limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no
  Security: disabled
  DHCPv4 Snooping: disabled
  MTU: 1500
  MAC Filter:  Static MAC addresses:
  ACs: 1 (1 up), VFIs: 1, PWs: 2 (1 up)
```

```
List of ACs:
  AC: GigabitEthernet0/4/0/1, state is up
    Type Ethernet
    MTU 1500; XC ID 0x5000001; interworking none; MSTi 0 (unprotected)
    MAC Learning: enabled
    MAC withdraw: disabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown Unicast: enabled
    MAC address aging time: 300 s Type: inactivity
    MAC address limit: 4000, Action: none, Notification: syslog
    MAC limit reached: no
    Security: disabled
    DHCPv4 Snooping: disabled
    Static MAC addresses:
    Statistics:
      packet totals: receive 6,send 0
      byte totals: receive 360,send 4
List of Access PWs:
List of VFIs:
  VFI siva_vfi
    PW: neighbor 1.1.1.1, PW ID 1, state is down ( local ready )
      PW class not set, XC ID 0xff000001
      Encapsulation MPLS, protocol LDP
      PW type Ethernet, control word enabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
             MPLS         Local                          Remote
        ------------ ------------------------------ -------------------------
             Label          30005                          unknown
             Group ID     0x0                              0x0
             Interface    siva/vfi                         unknown
             MTU          1500                             unknown
             Control word enabled                       unknown
             PW type      Ethernet                        unknown
        ------------ ------------------------------ -------------------------
      Create time: 19/11/2007 15:20:14 (00:25:25 ago)
      Last time status changed: 19/11/2007 15:44:00 (00:01:39 ago)
      MAC withdraw message: send 0 receive 0
```

Configuring the MAC Address Limit

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **limit**
7. **maximum** {*value*}
8. **action** {**flood** | **no-flood** | **shutdown**}
9. **notification** {**both** | **none** | **trap**}

10. **end**
    or
    commit
11. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6	**limit**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# limit`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)#`
		Sets the MAC address limit for action, maximum, and notification and enters L2VPN bridge group bridge domain MAC limit configuration mode.

Step 7	**maximum** {*value*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# maximum 5000`	Configures the specified action when the number of MAC addresses learned on a bridge is reached.

	Command or Action	Purpose
Step 8	**action** {**flood** | **no-flood** | **shutdown**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# action flood`	Configures the bridge behavior when the number of learned MAC addresses exceed the MAC limit configured.
Step 9	**notification** {**both** | **none** | **trap**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# notification both`	Specifies the type of notification that is sent when the number of learned MAC addresses exceeds the configured limit.

Step 10	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 11	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details about the MAC address limit.

Configuring the MAC Address Aging

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*

4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **aging**
7. **time** {*seconds*}
8. **type** {**absolute** | **inactivity**}
9. **end**
   or
   commit
10. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6	**aging**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# aging`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#`
		Enters the MAC aging configuration submode to set the aging parameters such as time and type.

	Command or Action	Purpose
Step 7	**time** {*seconds*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# time 300`	Configures the maximum aging time. • Use the *seconds* argument to specify the maximum age of the MAC address table entry. The range is from 300 to 30000 seconds. Aging time is counted from the last time that the switch saw the MAC address. The default value is 300 seconds.
Step 8	**type** {**absolute** | **inactivity**} **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# type absolute`	Configures the type for MAC address aging. • Use the **absolute** keyword to configure the absolute aging type. • Use the **inactivity** keyword to configure the inactivity aging type.

Step 9	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 10	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details about the aging fields.

Disabling MAC Flush at the Bridge Port Level

Perform this task to disable the MAC flush at the bridge domain level.

You can disable the MAC flush at the bridge domain, bridge port or access pseudowire levels. By default, the MACs learned on a specific port are immediately flushed, when that port becomes nonfunctional.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **port-down flush disable**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/CPU0:router(config-l2vpn-bg)#`
		Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4	**bridge-domain** *bridge-domain-name*
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**mac**
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters l2vpn bridge group bridge domain MAC configuration mode.

	Command or Action	Purpose
Step 6	**port-down flush disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# port-down flush disable`	Disables MAC flush when the bridge port becomes nonfunctional.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Multisegment Pseudowire

This section describes these tasks:

Provisioning a Multisegment Pseudowire Configuration

Configure a multisegment pseudowire as a point-to-point (p2p) cross-connect. For more information refer, Figure 15 on page 96. Here, the **xconnect group** item corresponds to the MPLS/IP. The **neighbor** item corresponds to the destination PE node with its IP address and the **pw-id**.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **neighbor** *A.B.C.D* **pw-id** *value*
6. **pw-class** *class-name*
7. **exit**
8. **neighbor** *A.B.C.D* **pw-id** *value*
9. **pw-class** *class-name*
10. **commit**

DETAILED STEPS

	Command	Purpose

Step 1	`configure`
	
	Example:
	`RP/0/RSP0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`l2vpn`
	
	Example:
	`RP/0/RSP0/CPU0:router(config)# l2vpn`
		Enters Layer 2 VPN configuration mode.

	Command	Purpose
Step 3	**xconnect group** *group-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group MS-PW1`	Configures a cross-connect group name using a free-format 32-character string.
Step 4	**p2p** *xconnect-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p ms-pw1`	Enters P2P configuration submode.
Step 5	**neighbor** *A.B.C.D* **pw-id** *value* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.165.200.25 pw-id 100`	Configures a pseudowire for a cross-connect. The IP address is that of the corresponding PE node. The **pw-id** must match the **pw-id** of the PE node. **Note** The psuedowire configuration is done on an S-PE node.
Step 6	**pw-class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls`	Enters pseudowire class submode, allowing you to define a pseudowire class template.
Step 7	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# exit`	Exits pseudowire class submode and returns the router to the parent configuration mode.
Step 8	**neighbor** *A.B.C.D* **pw-id** *value* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.165.202.158 pw-id 300`	Configures a pseudowire for a cross-connect. The IP address is that of the corresponding PE node. The **pw-id** must match the **pw-id** of the PE node. **Note** The psuedowire configuration is done on an S-PE node.
Step 9	**pw-class** *class-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls`	Enters pseudowire class submode, allowing you to define a pseudowire class template.
Step 10	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit`	Saves configuration changes to the running configuration file and remains in the configuration session.

Provisioning a Global Multisegment Pseudowire Description

S-PE nodes must have a description in the Pseudowire Switching Point Type-Length-Value (TLV). The TLV records all the switching points the pseudowire traverses, creating a helpful history for troubleshooting. For more information refer, Figure 16 on page 97.

Each multisegment pseudowire can have its own description. For instructions, see the "Provisioning a Cross-Connect Description" section on page 143. If it does not have one, this global description is used.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **description** *value*
4. **commit**

DETAILED STEPS

	Command	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/RSP0/CPU0:router(config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 3	**description** *value* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn)# description S-PE1`	Populates the Pseudowire Switching Point TLV. This TLV records all the switching points the pseudowire traverses. Each multisegment pseudowire can have its own description. If it does not have one, this global description is used. **Note** The psuedowire configuration is done on all S-PE nodes.
Step 4	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn)# commit`	Saves configuration changes to the running configuration file and remains in the configuration session.

Provisioning a Cross-Connect Description

S-PE nodes must have a description in the Pseudowire Switching Point TLV. The TLV records all the switching points the pseudowire traverses, creating a history that is helpful for troubleshooting.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **description** *value*
6. **commit**

DETAILED STEPS

	Command	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/RSP0/CPU0:router(config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 3	**xconnect group** *group-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group MS-PW1`	Configures a cross-connect group name using a free-format 32-character string.
Step 4	**p2p** *xconnect-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p ms-pw1`	Enters P2P configuration submode.

	Command	Purpose
Step 5	**description** *value* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# description MS-PW from T-PE1 to T-PE2`	Populates the Pseudowire Switching Point TLV. This TLV records all the switching points the pseudowire traverses. Each multisegment pseudowire can have its own description. If it does not have one, a global description is used. For more information, see the "Provisioning a Multisegment Pseudowire Configuration" section on page 140. **Note** The psuedowire configuration is done on all S-PE nodes.
Step 6	**commit** **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# commit`	Saves configuration changes to the running configuration file and remains in the configuration session.

Provisioning Switching Point TLV Security

For security purposes, the TLV can be hidden, preventing someone from viewing all the switching points the pseudowire traverses.

Virtual Circuit Connection Verification (VCCV) may not work on multisegment pseudowires with the **switching-tlv** parameter set to "hide".

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation mpls**
5. **protocol ldp**
6. **switching-tlv hide**
7. **commit**

DETAILED STEPS

	Command	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/RSP0/CPU0:router (config)# l2vpn`	Enters Layer 2 VPN configuration mode.

	Command	Purpose
Step 3	**pw-class** *class-name* **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn)# pw-class dynamic_mpls`	Enters pseudowire class submode, allowing you to define a pseudowire class template.
Step 4	**encapsulation mpls** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation mpls`	Sets pseudowire encapsulation to MPLS.
Step 5	**protocol ldp** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# protocol ldp`	Sets pseudowire signaling protocol to LDP.
Step 6	**switching-tlv hide** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# switching-tlv hide`	Sets pseudowire TLV to hide. **Note** The psuedowire configuration is done on all S-PE nodes.
Step 7	**commit** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# commit`	Saves configuration changes to the running configuration file and remains in the configuration session.

Enabling Multisegment Pseudowires

Use the **pw-status** command after you enable the **pw-status** command. The **pw-status** command is disabled by default. Changing the **pw-status** command reprovisions all pseudowires configured under L2VPN.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-status**
4. **commit**

DETAILED STEPS

	Command	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/RSP0/CPU0:router (config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 3	**pw-status** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn)# pw-status`	Enables all pseudowires configured on this Layer 2 VPN. **Note** Use the **pw-status disable** command to disable pseudowire status.
Step 4	**commit** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn)# commit`	Saves configuration changes to the running configuration file and remains in the configuration session.

Configuring Pseudowire Redundancy

Pseudowire redundancy allows you to configure a backup pseudowire in case the primary pseudowire fails. When the primary pseudowire fails, the PE router can switch to the backup pseudowire. You can elect to have the primary pseudowire resume operation after it comes back up.

These topics describe how to configure pseudowire redundancy:

- Configuring a Backup Pseudowire, page VPC-146
- Configuring Point-to-Point Pseudowire Redundancy, page VPC-149
- Forcing a Manual Switchover to the Backup Pseudowire, page VPC-151

Configuring a Backup Pseudowire

Perform this task to configure a backup pseudowire for a point-to-point neighbor.

Note When you reprovision a primary pseudowire, traffic resumes in two seconds. However, when you reprovision a backup pseudowire, traffic will resume after a delay of 45 to 60 seconds. This is the expected behavior.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*

4. **p2p** {*xconnect-name*}
5. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
6. **backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/RSP0/CPU0:router(config)# l2vpn`
	`RP/0/RSP0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**xconnect group** *group-name*
	
	Example:
	`RP/O/RSP0/CPU0:router(config-l2vpn)# xconnect group A`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc)#`
		Enters the name of the cross-connect group.

Step 4	**p2p** {*xconnect-name*}
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p xc1`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

Step 5	**neighbor** {*A.B.C.D*} {**pw-id** *value*} **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 2`	Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose

Step 6	**backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#`
		Configures the backup pseudowire for the cross-connect.
		• Use the **neighbor** keyword to specify the peer to cross-connect. The IP address argument (*A.B.C.D*) is the IPv4 address of the peer.
		• Use the **pw-id** keyword to configure the pseudowire ID. The range is from 1 to 4294967295.

Step 7	**end**
	or
	commit
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# end`
	or
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Point-to-Point Pseudowire Redundancy

Perform this task to configure point-to-point pseudowire redundancy for a backup delay.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** {*class-name*}
4. **backup disable** {**delay** *value* | **never**}
5. **exit**
6. **xconnect group** *group-name*
7. **p2p** {*xconnect-name*}
8. **neighbor** {*A.B.C.D*} {**pw-id** *value*}
9. **pw-class** {*class-name*}
10. **backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/RSP0/CPU0:router(config)# l2vpn`
	`RP/0/RSP0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**pw-class** {class-*name*}
	
	Example:
	`RP/O/RSP0/CPU0:router(config-l2vpn)# pw-class path1`
	`RP/0/RSP0/CPU0:router(config-l2vpn-pwc)#`
		Configures the pseudowire class name.

	Command or Action	Purpose
Step 4	**backup disable** {**delay** *value* | **never**} **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20`	This command specifies how long the primary pseudowire should wait after it becomes active to take over for the backup pseudowire. • Use the **delay** keyword to specify the number of seconds that elapse after the primary pseudowire comes up before the secondary pseudowire is deactivated. The range, in seconds, is from 0 to 180. • Use the **never** keyword to specify that the secondary pseudowire does not fall back to the primary pseudowire if the primary pseudowire becomes available again, unless the secondary pseudowire fails.

Step 5	**exit**
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# exit`
	`RP/O/RSP0/CPU0:router(config-l2vpn)#`
		Exits the current configuration mode.

Step 6	**xconnect group** *group-name*
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group A`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc)#`
		Enters the name of the cross-connect group.

Step 7	**p2p** {*xconnect-name*}
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p xc1`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

Step 8	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 2`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire segment for the cross-connect.

Step 9	**pw-class** {class-*name*} **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class path1`	Configures the pseudowire class name.

	Command or Action	Purpose

Step 10	**backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#`
		Configures the backup pseudowire for the cross-connect.
		• Use the **neighbor** keyword to specify the peer to the cross-connect. The A.B.C.D argument is the IPv4 address of the peer.
		• Use the **pw-id** keyword to configure the pseudowire ID. The range is from 1 to 4294967295.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# end`
	or
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Forcing a Manual Switchover to the Backup Pseudowire

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

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

Configuring Pseudowire Headend

The PWHE is created by configuring interface pw-ether or pw-iw. For the PWHE to be functional, the xconnect has to be configured completely. Configuring other layer 3 (L3) parameters, such as VRF and IP addresses, are optional for the PWHE to be functional. However, the L3 features are required for the layer 3 services to be operational; that is, for PW L3 termination.

This section describes these topics:

PWHE Configuration Restrictions

These are the configuration restrictions for PWHE:

- Up to 3600 PWHE interfaces (a combination of pw-ether and pw-iw).
- Up to eight interface lists per peer.
- Up to four L3 links per interface list.
- VLAN ID (tag-impose) can be configured only in xconnects which have pw-ether interfaces.
- VLAN ID (tag-impose) can only be configured under VC type 4 pw-ether interfaces.
- Interface lists can accept POS, GigabitEthernet, TenGigabitEthernet; other interfaces are rejected.
- No support for features such as pseudowire redundancy, preferred path, local switching or L2TP for xconnects configured with PWHE.
- Ethernet and VLAN transport modes are not allowed for pw-iw xconnects.
- Address family, Cisco Discovery Protocol (CDP) and MPLS configurations are not allowed on PWHE interfaces.
- IPv6 configuration is not allowed under pw-iw interfaces.

Configuring PWHE Interfaces

Perform this task to configure PWHE interfaces.

Summary Steps

1. **configure**
2. **interface pw-ether** *id*
3. **attach generic-interface-list** *interface_list_name*
4. **end**
   or
   commit

Detailed Steps

	Command or Action	Purpose

Step 1	**configure**
	
	Example:
	`RP/0/0/CPU0:router# configure`
	`RP/0/0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	**interface pw-ether** *id* **Example:** `RP/0/0/CPU0:router(config)# interface pw-ether <id>`	Configures the PWHE interface and enters the interface configuration mode.
Step 3	**attach generic-interface-list** *interface_list_name* **Example:** `RP/0/0/CPU0:router(config-if)# attach generic-interface-list interfacelist1`	Attaches the interface to a specified interface list.

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Restrictions for Configuring PWHE Interfaces

These are the restrictions for configuring PWHE interfaces:

- Neighbor and pw-ID pair must be unique in L2VPN.
- pw-ether interfaces have to be VC type 4 or 5.
- pw-iw interfaces cannot have IPv6 address because IPv6 is not supported on pw-iw (VC type 11). The VC type is set to type 11 if AC is pw-iw even when interworking ipv4 is not configured.
- The VLAN ID is allowed only if VC type is 4.

- MPLS protocols (MPLS-TE, LDP, RSVP) cannot be configured on PW-HE.
- No interface list configuration is accepted on non-PWHE platforms.

Configuring PWHE Interface Parameters

Perform this task to configure PWHE interface parameters.

Summary Steps

1. **configure**
2. **interface pw-ether** *id*
3. **attach generic-interface-list** *interface_list_name*
4. **l2overhead** *bytes*
5. **load-interval** *seconds*
6. **dampening** *decay-life*
7. **logging events link-status**
8. **mac-address** *MAC address*
9. **mtu** *interface_MTU*
10. **bandwidth** *bandwidth*
11. **end**
    or
    commit

Detailed Steps

	Command or Action	Purpose

Step 1	**configure**
	
	Example:
	`RP/0/0/CPU0:router# configure`
	`RP/0/0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	**interface pw-ether** *id* **Example:** `RP/0/0/CPU0:router(config)# interface pw-ether <id>`	Configures the PWHE interface and enters the interface configuration mode.
Step 3	**attach generic-interface-list** *interface_list_name* **Example:** `RP/0/0/CPU0:router(config-if)# attach generic-interface-list interfacelist1`	Attaches the interface to a specified interface list.
Step 4	**l2overhead** *bytes* **Example:** `RP/0/0/CPU0:router(config-if)#l2overhead 20`	Sets layer 2 overhead size.
Step 5	**load-interval** *seconds* **Example:** `RP/0/0/CPU0:router(config-if)#load-interval 90`	Specifies interval, in seconds, for load calculation for an interface. The number of seconds: • Can be set to 0 [0 disables load calculation] • If not 0, interval must be specified in multiples of 30 between 30 and 600.
Step 6	**dampening** *decay-life* **Example:** `RP/0/0/CPU0:router(config-if)#dampening 10`	Configures state dampening on the given interface (in minutes).
Step 7	**logging events link-status** **Example:** `RP/0/0/CPU0:router(config-if)#logging events link-status`	Configures per interface logging.
Step 8	**mac-address** *MAC address* **Example:** `RP/0/0/CPU0:router(config-if)#mac-address aaaa.bbbb.cccc`	Sets the MAC address (xxxx.xxxx.xxxx) on an interface.

	Command or Action	Purpose
Step 9	**mtu** *interface_MTU* **Example:** `RP/0/0/CPU0:router(config-if)#mtu 128`	Sets the MTU on an interface.
Step 10	**bandwidth** *bandwidth* **Example:** `RP/0/0/CPU0:router(config-if)#bandwidth 3987`	Sets the bandwidth of an interface.

Step 11	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Pseudowire Source Address

Source address is configured under pseudowire class with encapsulation set to MPLS. This enables flexible LDP target to support Rx pindown. Perform this task to configure the source IPv4 address.

Summary Steps

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation mpls**
5. **ipv4 source** *A.B.C.D*
6. **end**
   or
   commit

Detailed Steps

	Command or Action	Purpose

Step 1	**configure**
	
	Example:
	`RP/0/0/CPU0:router# configure`
	`RP/0/0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 3	**pw-class** *class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class class1`	Enters pseudowire class submode, allowing you to define a pseudowire class template.
Step 4	**encapsulation mpls** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls`	Sets pseudowire encapsulation to MPLS.
Step 5	**ipv4 source** *A.B.C.D* **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-mpls)# ipv4 source w-ether 100`	Sets the local source IPv4 address.

Step 6	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-mpls)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-mpls)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring PWHE Crossconnect

Perform this task to configure PWHE crossconnects.

Summary Steps

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface pw-ether** *id*
6. **neighbor** *A.B.C.D* **pw-id** *value*
7. **pw-class** *class-name*
8. **end**
   or
   commit

Detailed Steps

	Command or Action	Purpose

Step 1	**configure**
	
	Example:
	`RP/0/0/CPU0:router# configure`
	`RP/0/0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 3	**xconnect group** *group-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group MS-PW1`	Configures a cross-connect group name using a free-format 32-character string.
Step 4	**p2p** *xconnect-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p ms-pw1`	Enters P2P configuration submode.
Step 5	**interface pw-ether** *id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface pw-ether 100`	Configures the PWHE interface.
Step 6	**neighbor** *A.B.C.D* **pw-id** *value* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.165.200.25 pw-id 100`	Configures a pseudowire for a cross-connect. The IP address is that of the corresponding PE node. The **pw-id** must match the **pw-id** of the PE node.

	Command or Action	Purpose
Step 7	**pw-class** *class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls`	Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 8	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Flow Aware Transport Pseudowire

This section provides information on

- Enabling Load Balancing with ECMP and FAT PW for VPWS

Enabling Load Balancing with ECMP and FAT PW for VPWS

Perform this task to enable load balancing with ECMP and FAT PW for VPWS.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** {*name*}
4. **encapsulation mpls**
5. **load-balancing flow-label** {**both** | **receive** | **transmit**} [**static**]
6. **exit**
7. **xconnect group** *group-name*
8. **p2p** *xconnect-name*
9. **interface type** *interface-path-id*
10. **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*
11. **pw-class** {*name*}
12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.
Step 3	**pw-class** {*name*} **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class path1`	Configures the pseudowire class template name to use for the pseudowire.
Step 4	**encapsulation mpls** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls`	Configures the pseudowire encapsulation to MPLS.

	Command or Action	Purpose
Step 5	**load-balancing flow-label** {**both** | **receive** | **transmit**} [**static**] **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-mpls)# load-balancing flow-label both`	Enables load-balancing on ECMPs. Also, enables the imposition and disposition of flow labels for the pseudowire. **Note** If the **static** keyword is not specified, end to end negotiation of the FAT PW is enabled.
Step 6	**exit** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-mpls)#exit`	Exits the pseudowire encapsulation submode and returns the router to the parent configuration mode.
Step 7	**xconnect group** *group-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp1`	Specifies the name of the cross-connect group.
Step 8	**p2p** *xconnect-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p vlan1`	Specifies the name of the point-to-point cross-connect
Step 9	**interface type** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1`	Specifies the interface type and instance.
Step 10	**neighbor** *A.B.C.D* **pw-id** *pseudowire-id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.2.2.2 pw-id 2000`	Configures the pseudowire segment for the cross-connect. Use the A.B.C.D argument to specify the IP address of the cross-connect peer. **Note** A.B.C.D can be a recursive or non-recursive prefix.

	Command or Action	Purpose
Step 11	**pw-class** *class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class path1`	Associates the pseudowire class with this pseudowire.

Step 12	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	
	or
	
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling Pseudowire Grouping

Perform this task to enable pseudowire grouping.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-grouping**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters configuration mode.

Step 2	**l2vpn**
	
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

	Command or Action	Purpose
Step 3	**pw-grouping** **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-grouping`	Enables pseudowire grouping

Step 4	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn)# commit`
		Saves configuration changes.
		• When you issue the **end** command, the system prompts you to commit changes:
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		– Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
		– Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
		– Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Virtual Private LAN Services

This section includes the following configuration examples:

Virtual Private LAN Services Configuration for Provider Edge-to-Provider Edge: Example

These configuration examples show how to create a Layer 2 VFI with a full-mesh of participating VPLS provider edge (PE) nodes.

The following configuration example shows how to configure PE 1:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE1-VPLS-A
    GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 2.2.2.2 pw-id 1---PW1
     neighbor 3.3.3.3 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 1.1.1.1 255.255.255.25
  commit
```

The following configuration example shows how to configure PE 2:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE2-VPLS-A
    interface GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 1.1.1.1 pw-id 1---PW1
     neighbor 3.3.3.3 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 2.2.2.2 255.255.255.25
  commit
```

The following configuration example shows how to configure PE 3:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE3-VPLS-A
    interface GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 1.1.1.1 pw-id 1---PW1
     neighbor 2.2.2.2 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 3.3.3.3 255.255.255.25
  commit
```

Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example

The following configuration shows how to configure VPLS for a PE-to-CE nodes:

```
configure
 interface GigabitEthernet0/0
  l2transport---AC interface
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end

configure
 interface GigabitEthernet0/0
  l2transport
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end

configure
 interface GigabitEthernet0/0
  l2transport
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
```

Configuring Backup Disable Delay: Example

The following example shows how a backup delay is configured for point-to-point PW where the backup disable delay is 50 seconds:

```
l2vpn
pw-class class_1
backup disable delay 20
exit
xconnect group_A
p2p rtrX_to_rtrY
neighbor 1.1.1.1 pw-id 2
pw-class class_1
backup neighbor 2.2.2.2 pw- id 5
commit
```

The following example shows how a backup delay is configured for point-to-point PW where the backup disable delay is never:

```
l2vpn
pw-class class_1
backup disable  never
exit
```

```
xconnect group_A
p2p rtrX_to_rtrY
     neighbor 1.1.1.1 pw-id 2
pw-class class_1
     backup neighbor 2.2.2.2 pw-id 5
commit
```

Blocking Unknown Unicast Flooding: Example

Unknown-unicast flooding can be blocked at the following levels:

- bridge domain
- bridge port (attachment circuit (AC))
- access pseudowire (PW)

The following example shows how to block unknown-unicast flooding at the bridge domain level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    flooding unknown-unicast disable
  end
```

The following example shows how to block unknown-unicast flooding at the bridge port level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    interface POS 0/1/0/1
    flooding unknown-unicast disable
  end
```

The following example shows how to block unknown-unicast flooding at the access pseudowire level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    neighbor 10.1.1.1 pw-id 1000
    flooding unknown-unicast disable
  end
```

Disabling MAC Flush: Examples

You can disable the MAC flush at the following levels:

- bridge domain
- bridge port (attachment circuit (AC))
- access pseudowire (PW)

The following example shows how to disable the MAC flush at the bridge domain level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    mac
    port-down flush disable
```

```
  end
```

The following example shows how to disable the MAC flush at the bridge port level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    interface POS 0/1/0/1
    mac
    port-down flush disable
  end
```

The following example shows how to disable the MAC flush at the access pseudowire level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    neighbor 10.1.1.1 pw-id 1000
    mac
    port-down flush disable
   end
```

H-VPLS with QinQ or QinAny: Examples

This example shows the QinQ or QinAny AC type in the output of the **show l2vpn forwarding bridge-domain hardware ingress/egress** command:

```
INGRESS AC [version, state]: [1, BOUND]

        Xconnect-ID: [15] TCAM-Key: (UIDB:0x4 O-vlan:2 I-vlan:2 Ether-Type:0x8100)
        HW: 0x34001000 0x0118000f 0x1011801c 0xc7ff5100
        SW: 0x34001000 0x0118000f 0x1011801c 0xc7ff5100

        Service type: 7 (bridging pmp QinQ)
        Entry type: 1 (fwd)
        Bridge_ID : 0
        ACL_ID : 4096
        Xconnect_ID : 0x118000f
        SplitHorizonGroup_ID : 0
        Rewrite supported: 0 (No)
        PW_mode: 0 (vc-type 5)
        AC-type: 1 (qinq-mode)
        Interface handle: 0x11801c
        Ingress AC stats: 0x7ff51

   EGRESS AC [version, state]: [1, BOUND]

        Xconnect-ID: [15] TLU2-entry-addr: [0x200a00f]
        HW: 0x8018b000 0x0000000f 0x00004002 0xfb748000
        SW: 0x8018b000 0x0000000f 0x00004002 0xfb748000

        Entry status: 1 (Fwd)
        AC_type: 1 (qinq-mode)
        Outer-vlan: 2
        Inner-vlan: 2
        Outer Ether Type: 0 (dot1q)
        AC_mtu: 1580
        Adjacency_type: 3
```

```
        Default EgressQ (SharqQ): 15
        PW mode: 0 (vc-type 5)
        Rewrite supported: 0 (No)
        Control-word supported: 0 (No)
        Egress AC stats: 0x7dba4
```

H-VPLS with Access-PWs: Examples

This example shows the PW type in the output of the **show l2vpn forwarding bridge-domain hardware ingress/egress** command:

```
Ingress:
  INGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: Access PW
        XID: 127/15/CPU0 : 1 (0xfff80001)
        Bridge ID: 0, Split Horizon ID: 0
        VC label: 16010

  INGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: VFI(Core) PW
        XID: 127/15/CPU0 : 2 (0xfff80002)
        Bridge ID: 0, Split Horizon ID: 1
        VC label: 16007

Egress:
        OIF[1] seg_type: Access PW xid: 0xfff80001 ecd_ptr: 0x500a
        TLU RESULT tlu_addr: 0x200bc00 ch: 2 seg_type: 0
        HW: 0x0000500a 0x00000000 0xfff80001 0x03e8a000
        SW: 0x0000500a 0x00000000 0xfff80001 0x03e8a000
        SHG: 0
        XID: 0xfff80001
…
        OIF[2] seg_type: VFI(Core) PW xid: 0xfff80002 ecd_ptr: 0x500f
        TLU RESULT tlu_addr: 0x3000601 ch: 3 seg_type: 0
        HW: 0x0100500f 0x00000000 0xfff80002 0x03e87000
        SW: 0x0100500f 0x00000000 0xfff80002 0x03e87000
        SHG: 1
        XID: 0xfff80002
…
  EGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: Access PW
        XID: 127/15/CPU0 : 1 (0xfff80001)
        Bridge ID: 0, Split Horizon ID: 0
        VC label: 16010
…
  EGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: VFI(Core) PW
        XID: 127/15/CPU0 : 2 (0xfff80002)
        Bridge ID: 0, Split Horizon ID: 1
        VC label: 16007
```

Pseudowires: Examples

The examples include these devices and connections:

- T-PE1 node has:
  - Cross-connect with an AC interface (facing CE1)
  - Pseudowire to S-PE1 node

  - IP address 209.165.200.225
- T-PE2 node
  - Cross-connect with an AC interface (facing CE2)
  - Pseudowire to S-PE1 node
  - IP address 209.165.200.254
- S-PE1 node
  - Multisegment pseudowire cross-connect with a pseudowire segment to T-PE1 node
  - Pseudowire segment to T-PE2 node
  - IP address 209.165.202.158

Configuring Dynamic Pseudowires at T-PE1 Node: Example

```
RP/0/RSP0/CPU0:T-PE1# configure
RP/0/RSP0/CPU0:T-PE1(config)# l2vpn
RP/0/RSP0/CPU0:T-PE1 (config-l2vpn)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:T-PE1(config-l2vpn)# xconnect group XCON1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# description T-PE1 MS-PW to 10.165.202.158
via 10.165.200.254
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/0.1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 100
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Dynamic Pseudowires at S-PE1 Node: Example

```
RP/0/RSP0/CPU0:S-PE1# configure
RP/0/RSP0/CPU0:S-PE1(config)# l2vpn
RP/0/RSP0/CPU0:S-PE1(config-l2vpn)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn)# xconnect group MS-PW1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc)# p2p ms-pw1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# description S-PE1 MS-PW between
10.165.200.225 and 10.165.202.158
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.200.225 pw-id 100
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
```

```
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.202.158 pw-id 300
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Dynamic Pseudowires at T-PE2 Node: Example

```
RP/0/RSP0/CPU0:T-PE2# configure
RP/0/RSP0/CPU0:T-PE2(config)# l2vpn
RP/0/RSP0/CPU0:T-PE2 (config-l2vpn)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE2 (config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:T-PE2 (config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:T-PE2 (config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:T-PE2 (config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:T-PE2 (config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:T-PE2(config-l2vpn)# xconnect group XCON1
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# description T-PE2 MS-PW to 10.165.200.225 via
10.165.200.254
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# interface gigabitethernet 0/2/0/0.4
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 300
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Dynamic Pseudowires and Preferred Paths at T-PE1 Node: Example

```
RP/0/RSP0/CPU0:T-PE1# configure
RP/0/RSP0/CPU0:T-PE1(config)# l2vpn
RP/0/RSP0/CPU0:T-PE1(config-l2vpn)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te
1000
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:T-PE1(config-l2vpn)# xconnect group XCON1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# description T-PE1 MS-PW to 10.165.202.158
via 10.165.200.254
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/0.1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 100
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Dynamic Pseudowires and Preferred Paths at S-PE1 Node: Example

```
RP/0/RSP0/CPU0:S-PE1# configure
RP/0/RSP0/CPU0:S-PE1(config)# l2vpn
RP/0/RSP0/CPU0:S-PE1(config-l2vpn)# pw-class dynamic_mpls1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te
1000
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn)# pw-class dynamic_mpls2
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te
2000
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn)# xconnect group MS-PW1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc)# p2p ms-pw1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# description S-PE1 MS-PW between
10.165.200.225 and 10.165.202.158
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.200.225 pw-id 100
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.202.158 pw-id 300
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls2
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Dynamic Pseudowires and Preferred Paths at T-PE2 Node: Example

```
RP/0/RSP0/CPU0:T-PE2# configure
RP/0/RSP0/CPU0:T-PE2(config)# l2vpn
RP/0/RSP0/CPU0:T-PE2(config-l2vpn)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te
2000
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc-encap-mpls)# exit
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:T-PE2(config-l2vpn)# xconnect group XCON1
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# description T-PE2 MS-PW to 10.165.200.225 via
10.165.200.254
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# interface gigabitethernet 0/2/0/0.4
```

```
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 300
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Static Pseudowires at T-PE1 Node: Example

```
RP/0/RSP0/CPU0:T-PE1# configure
RP/0/RSP0/CPU0:T-PE1(config)# l2vpn
RP/0/RSP0/CPU0:T-PE1(config-l2vpn)# xconnect group XCON1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/0.1
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 100
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p-pw)# mpls static label local 50 remote 400
RP/0/RSP0/CPU0:T-PE1(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Static Pseudowires at S-PE1 Node: Example

```
RP/0/RSP0/CPU0:S-PE1# configure
RP/0/RSP0/CPU0:S-PE1(config)# l2vpn
RP/0/RSP0/CPU0:S-PE1(config-l2vpn)# xconnect group MS-PW1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc)# p2p ms-pw1
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.200.225 pw-id 100
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# mpls static label local 400 remote 50
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# exit
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p)# neighbor 10.165.202.158 pw-id 300
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# mpls static label local 40 remote 500
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Static Pseudowires at T-PE2 Node: Example

```
RP/0/RSP0/CPU0:T-PE2# configure
RP/0/RSP0/CPU0:T-PE2(config)# l2vpn
RP/0/RSP0/CPU0:T-PE2(config-l2vpn)# xconnect group XCON1
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# interface gigabitethernet 0/2/0/0.4
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p)# neighbor 10.165.200.254 pw-id 300
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p-pw)# mpls static label local 500 remote 40
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-xc-p2p-pw)# commit
```

Configuring Multisegment Pseudowires: Examples

This example shows how to configure a multisegment pseudowire:

```
configure
    l2vpn
        xconnect group MS-PW1
        p2p ms-pw1
        neighbor 10.165.200.25 pw-id 100
```

```
    pw-class dynamic_mpls
    exit
    neighbor 10.165.202.158 pw-id 300
    pw-class dynamic_mpls
  end
```

show l2vpn xconnect

```
RP/0/RSP0/CPU0:router# show l2vpn xconnect
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        LU = Local Up, RU = Remote Up, CO = Connected

XConnect                   Segment 1                   Segment 2
Group      Name       ST   Description            ST   Description            ST
------------------------   ------------------------   ------------------------
MS-PW1     ms-pw1     UP   70.70.70.70     100    UP   90.90.90.90     300    UP
------------------------------------------------------------------------------
```

show l2vpn xconnect detail

```
RP/0/RSP0/CPU0:router# show l2vpn xconnect detail
Group MS-PW1, XC ms-pw1, state is up; Interworking none
  PW: neighbor 70.70.70.70, PW ID 100, state is up ( established )
    PW class not set
    Encapsulation MPLS, protocol LDP
    PW type Ethernet VLAN, control word enabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    PW Status TLV in use
        MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        16004                          16006
      Group ID     0x2000400                      0x2000700
      Interface    GigabitEthernet0/1/0/2.2       GigabitEthernet0/1/0/0.3
      MTU          1500                           1500
      Control word enabled                        enabled
      PW type      Ethernet VLAN                  Ethernet VLAN
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x5                            0x7
                   (control word)                 (control word)
                                                  (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------

    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing PW Switching TLVs (Label Mapping message):
      Local IP Address: 80.80.80.80, Remote IP address: 90.90.90.90, PW ID: 300
      Description: S-PE1 MS-PW between 70.70.70.70 and 90.90.90.90
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Statistics:
      packet totals: receive 0
      byte totals: receive 0
    Create time: 04/04/2008 23:18:24 (00:01:24 ago)
    Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
  PW: neighbor 90.90.90.90, PW ID 300, state is up ( established )
    PW class not set
    Encapsulation MPLS, protocol LDP
    PW type Ethernet VLAN, control word enabled, interworking none
```

```
PW backup disable delay 0 sec
Sequencing not set
PW Status TLV in use
    MPLS         Local                          Remote
  ------------ ------------------------------ ------------------------------
  Label        16004                          16006
  Group ID     0x2000800                      0x2000200
  Interface    GigabitEthernet0/1/0/0.3       GigabitEthernet0/1/0/2.2
  MTU          1500                           1500
  Control word enabled                        enabled
  PW type      Ethernet VLAN                  Ethernet VLAN
  VCCV CV type 0x2                            0x2
               (LSP ping verification)        (LSP ping verification)
  VCCV CC type 0x5                            0x7
               (control word)                 (control word)
                                              (router alert label)
               (TTL expiry)                   (TTL expiry)
  ------------ ------------------------------ ------------------------------
Incoming Status (PW Status TLV):
  Status code: 0x0 (Up) in Notification message
Outgoing PW Switching TLVs (Label Mapping message):
  Local IP Address: 80.80.80.80, Remote IP address: 70.70.70.70, PW ID: 100
  Description: S-PE1 MS-PW between 70.70.70.70 and 90.90.90.90
Outgoing Status (PW Status TLV):
  Status code: 0x0 (Up) in Notification message
Statistics:
  packet totals: receive 0
  byte totals: receive 0
Create time: 04/04/2008 23:18:24 (00:01:24 ago)
Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
```

show l2vpn xconnect summary

```
RP/0/RSP0/CPU0:router# show l2vpn xconnect summary
Number of groups: 1
Number of xconnects: 1
  Up: 1  Down: 0  Unresolved: 0
  AC-PW: 0  AC-AC: 0  PW-PW: 1
Number of Admin Down segments: 0
```

Configuring Pseudowire Redundancy: Examples

This example shows how to configure a backup pseudowire for a point-to-point neighbor:

```
configure
    l2vpn
        xconnect group A
        p2p xc1
        neighbor 10.1.1.2 pw-id 2
        backup neighbor 10.2.2.2 pw-id 5
    end
```

show l2vpn xconnect

```
RP/0/RSP0/CPU0:router# show l2vpn xconnect

Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        LU = Local Up, RU = Remote Up, CO = Connected, SB = Standby
```

```
XConnect                    Segment 1                    Segment 2
Group      Name       ST    Description            ST    Description            ST
------------------------    --------------------------   --------------------------
g1         pw2        UP    Gi0/2/0/0.2            UP    110.110.110.110 2      UP
                                                         Backup
                                                         130.130.130.130 2      SB
------------------------------------------------------------------------------------
```

show l2vpn xconnect detail

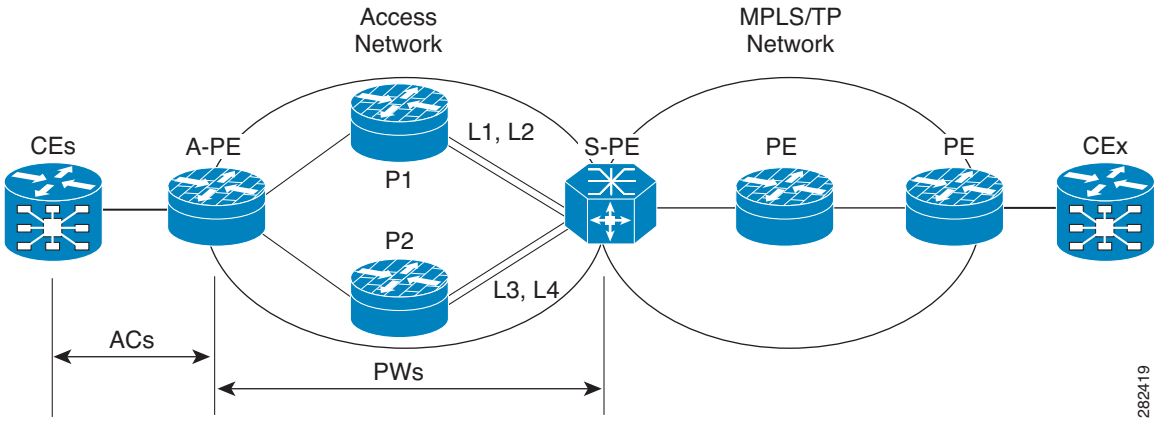
```
RP/0/RSP0/CPU0:router# show l2vpn xconnect detail
Group MS-PW1, XC ms-pw1, state is up; Interworking none
  PW: neighbor 10.165.200.225, PW ID 100, state is up ( established )
    PW class not set
    Encapsulation MPLS, protocol LDP
    PW type Ethernet VLAN, control word enabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    PW Status TLV in use
        MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        16004                          16006
      Group ID     0x2000400                      0x2000700
      Interface    GigabitEthernet0/1/0/2.2       GigabitEthernet0/1/0/0.3
      MTU          1500                           1500
      Control word enabled                        enabled
      PW type      Ethernet VLAN                  Ethernet VLAN
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x5                            0x7
                   (control word)                 (control word)
                                                  (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
    Incoming PW Switching TLVs (Label Mapping message):
      None
    Incoming Status (PW Status TLV and accompanying PW Switching TLV):
      Status code: 0x0 (no fault) in Notification message
    Outgoing PW Switching TLVs (Label Mapping message):
      Local IP Address: 10.165.200.254 , Remote IP address: 10.165.202.158 , PW ID: 300
      Description: S-PE1 MS-PW between 10.165.200.225 and 10.165.202.158
    Outgoing Status (PW Status TLV and accompanying PW Switching TLV):
      Status code: 0x0 (no fault) in Notification message
      Local IP Address: 10.165.200.254
    Create time: 04/04/2008 23:18:24 (00:01:24 ago)
    Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
    Statistics:
      packet totals: receive 0
      byte totals: receive 0
  PW: neighbor 10.165.202.158 , PW ID 300, state is up ( established )
    PW class not set
    Encapsulation MPLS, protocol LDP
    PW type Ethernet VLAN, control word enabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    PW Status TLV in use
        MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        16004                          16006
        Group ID     0x2000800                      0x2000200
      Interface    GigabitEthernet0/1/0/0.3       GigabitEthernet0/1/0/2.2
      MTU          1500                           1500
```

```
      Control word enabled                        enabled
      PW type      Ethernet VLAN                  Ethernet VLAN
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x5                            0x7
                   (control word)                 (control word)
                                                  (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Create time: 04/02/2009 19:28:59 (00:21:04 ago)
    Last time status changed: 04/02/2009 19:46:12 (00:03:51 ago)
    MAC withdraw message: send 0 receive 0
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
Backup PW:
  PW: neighbor 130.130.130.130, PW ID 2, state is standby ( all ready )
    Backup for neighbor 110.110.110.110 PW ID 2 ( inactive )
    PW class dynamic_mpls, XC ID 0x3000002
    Encapsulation MPLS, protocol LDP
    PW type Ethernet, control word enabled, interworking none
    Sequencing not set
    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        16001                          16002
      Group ID     0x3000200                      0x4
      Interface    GigabitEthernet0/2/0/0.2       3
      MTU          1500                           1500
      Control word enabled                        enabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x7                            0x7
                    (control word)                 (control word)
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Create time: 04/02/2009 19:28:59 (00:21:04 ago)
    Last time status changed: 04/02/2009 19:46:12 (00:03:51 ago)
    MAC withdraw message: send 0 receive 0
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

Configuring Pseudowire Headend: Example

This section provides an example of pseudowire headend configuration.

Figure 20 PWHE Configuration Example



Consider the topology in the above figure.

1. There are many customer edge routers (CEs) connected to a A-PE (each CE is connected using 1 link).
2. There are two P routers between A-PE an S-PE in the access network.
3. S-PE is connected by two links to P1—links L1 and L2 (on two separate linecards on P1 and S-PE); for example, Gig0/1/0/0 and Gig0/2/0/0 respectively.
4. S-PE is connected by two links to P2—L3 and L4 (on two separate linecards on P2 and S-PE); for example, Gig0/1/0/1 and Gig0/2/0/1 respectively.
5. For each CE-APE link, a xconnect (AC-PW) is configured on the A-PE. The PWs are connected to S-PE; some PWs are connected to [L1 (Gig0/1/0/0), L4 (Gig0/2/0/1)] and others through [L2 (Gig0/1/0/1), L3 (Gig0/2/0/0)].
6. A-PE uses router-id 100.100.100.100 for routing and PW signaling.
7. The two router-ids on S-PE used for PW signaling are 111.111.111.111 and 112.112.112.112 (for Rx pin-down). 110.110.110.110 is the router-id assigned for routing.

CE Configuration

Consider two CEs connected using GigabitEthernet0/3/0/0 (CE1 and A-PE) and GigabitEthernet0/3/0/1 (CE2 and A-PE).

At CE1:

```
interface Gig0/3/0/0
 ipv4 address 10.1.1.1/24
router static
 address-family ipv4 unicast
  110.110.110.110 Gig0/3/0/0
  A.B.C.D/N 110.110.110.110
```

At CE2:

```
interface Gig0/3/0/1
 ipv4 address 10.1.2.1/24
router static
 address-family ipv4 unicast
  110.110.110.110 Gig0/3/0/1
  A.B.C.D/N 110.110.110.110
```

A-PE Configuration

At A-PE, one xconnect is configured for each CE connection. Here, CE connections are L2 links, which are in xconnects. Each xconnect has a pseudowire connected to S-PE, though connected to different neighbor addresses, depending on where the pseudowire is to be pin downed: [L1, L4] or [L2, L3].

```
interface Gig0/3/0/0
 l2transport
interface Gig0/3/0/1
 l2transport

l2vpn
 xconnect group pwhe
  p2p pwhe_spe_1
   interface Gig0/3/0/0
   neighbor 111.111.111.111 pw-id 1
  p2p pwhe_spe_2
   interface Gig0/3/0/1
   neighbor 112.112.112.112 pw-id 2
```

P Router Configuration

Static routes are required on P routers for Rx pindown on S-PE to force PWs configured with a specific address to be transported over certain links.

At P1:

```
router static
 address-family ipv4 unicast
  111.111.111.111 Gig0/1/0/0
  112.112.112.112 Gig0/2/0/0
```

At P2:

```
router static
 address-family ipv4 unicast
  111.111.111.111 Gig0/2/0/1
  112.112.112.112 Gig0/1/0/1
```

S-PE Configuration

At S-PE, two PWHE interfaces (one for each PW) is configured, and each uses a different interface list for Tx pin-down. (This must match the static configuration at P routers for Rx pin-down). Each PWHE has the PW connected to A-PE (The pw-id must match the pw-id at A-PE.)

```
generic-interface-list il1
 interface gig0/1/0/0
 interface gig0/2/0/0
generic-interface-list il2
 interface gig0/1/0/1
 interface gig0/2/0/1

interface pw-ether1
 ipv4 address 10.1.1.2/24
 attach generic-interface-list il1
interface pw-ether2
 ipv4 address 10.1.2.2/24
 attach generic-interface-list il2

l2vpn
 xconnect group pwhe
  p2p pwhe1
   interface pw-ether1
```

```
 neighbor 100.100.100.100 pw-id 1
p2p pwhe2
 interface pw-ether2
 neighbor 100.100.100.100 pw-id 2
```

Configuring Flow Aware Transport Pseudowire: Example

This sample configuration shows how to enable load balancing with FAT PW for VPWS.

```
l2vpn
pw-class class1
    encapsulation mpls
        load-balancing flow-label transmit
    !
 !
pw-class class2
    encapsulation mpls
        load-balancing flow-label both
   !

xconnect group group1
    p2p p1
        interface GigabitEthernet 0/0/0/0.1
        neighbor 1.1.1.1 pw-id 1
            pw-class class1
        !
    !
!
```

Enabling Pseudowire Grouping: Example

This example shows how to enable pseudowire grouping.

```
config
l2vpn
  pw-grouping
```

Additional References

For additional information related to implementing VPLS, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS VPLS-related commands	*MPLS Virtual Private LAN Services Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*

Related Topic	Document Title
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
Cisco CRS router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 4447	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*, April 2006
RFC 4448	*Encapsulation Methods for Transport of Ethernet over MPLS Networks*, April 2006

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport