Implementing Multipoint Layer 2 Services

This module provides the conceptual and configuration information for Multipoint Layer 2 Bridging Services, also called Virtual Private LAN Services (VPLS).

Note 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 and Layer 2 VPNs.

For Point to Point Layer 2 Services, see *Implementing Point to Point Layer 2 Services* chapter.

For descriptions of the commands listed in this module, see the "Related Documents" section.

Feature History for Implementing Multipoint Layer 2 Services

Release	Modification
Release 3.8.0	This feature was introduced. Support for the bridging functionality feature (VPLS based) and pseudowire redundancy was added.
Release 3.9.0	The following features were added: • Blocking unknown unicast flooding. • Disabling MAC flush.
Release 4.0	The following features were added: • H-VPLS with MPLS Access pseudowire • H-VPLS with Ethernet Access • MAC Address withdrawal
Release 4.0.1	Support for the BGP Autodiscovery with LDP Signaling feature was added.
Release 4.1.0	Support for Pseudowire Headend feature was added.
Release 4.2.0	Support was added for: • VPLS pseudowire on LDP over TE and Preferred TE path • VPLS with Traffic Engineering Fast Reroute (TE FRR)

Release	Modification
Release 4.2.1	Support was added for: • Pseudowire Headend on Cisco CRS-3 router • IPv6 packets over PWHE interfaces
Release 4.3.0	Support was added for the Pseudowire Grouping feature.

Prerequisites for Implementing Multipoint Layer 2 Services

Before you configure Multipoint Layer 2 Services, 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 Multipoint Layer 2 Services are directly mapped to a TE tunnel.

Restrictions for Implementing Multipoint Layer 2 Services

The following restrictions are listed for implementing Multipoint Layer 2 Services:

- 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 Cisco CRS-1 router supports multiple types of attachment circuits in a bridge domain.
- The line card requires ternary content addressable memory (TCAM) Carving configuration. The Cisco CRS-1 router however, does not require the 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.
- For the Engine 5 line card, version 1 of the Ethernet SPA does not support QinQ mode and QinAny mode.
- The CRS-X line card does not support bundle interfaces on multipoint layer 2 services.

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. The Cisco CRS-1 router supports only the Ethernet port mode and the 802.1q VLAN mode.

Information About Implementing Multipoint Layer 2 Services

To implement Multipoint Layer 2 Services, you should understand these concepts:

Multipoint Layer 2 Services Overview

Multipoint Layer 2 Services enable 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.

Note Multipoint Layer 2 services are also called as Virtual Private LAN Services.

Multipoint Layer 2 Services 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. It offers simple Virtual LAN services that include flooding broadcast, multicast, and unknown unicast frames that are received on a bridge. The Multipoint Layer 2 Services solution requires a full mesh of pseudowires that are established among provider edge (PE) routers. The Multipoint Layer 2 Services 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.

Multipoint Layer 2 Services require the provider edge device to be MPLS-capable. The Multipoint Layer 2 Services provides 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.

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.

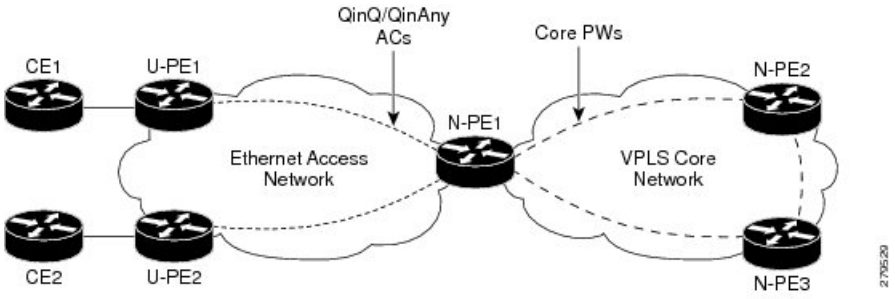
Two flavors of H-VPLS are:

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

H-VPLS with Ethernet Access QinQ or QinAny

The following figure shows the 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 1: Ethernet Access for H-VPLS

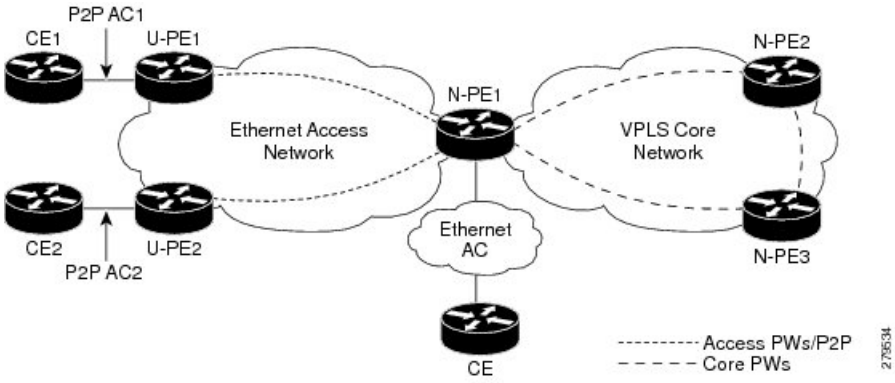


H-VPLS with PW-access

The following figure shows the 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 the below figure.

Figure 2: PW access for H-VPLS



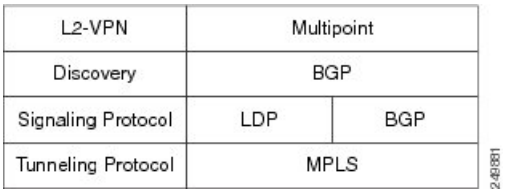
VPLS Discovery and Signaling

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

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

- VPLS Autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS Autodiscovery enables each VPLS PE router to discover the other provider edge (PE) routers that are part of the same VPLS domain.
- Once the PEs are discovered, pseudowires (PWs) are signaled and established across each pair of PE routers forming a full mesh of PWs across PE routers in a VPLS domain

Figure 3: VPLS Autodiscovery and Signaling



BGP-based VPLS Autodiscovery

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

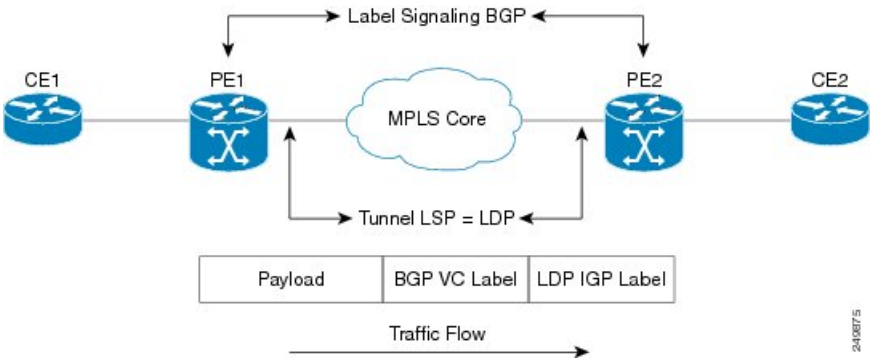
BGP-based VPLS autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS autodiscovery enables each VPLS PE router to discover the other provider edge (PE) routers that are part of the same VPLS domain. VPLS Autodiscovery also tracks when PE routers are added to or removed from the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

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

BGP Auto Discovery With BGP Signaling

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

Figure 4: Discovery and Signaling Attributes



The BGP signaling and autodiscovery scheme has the following components:

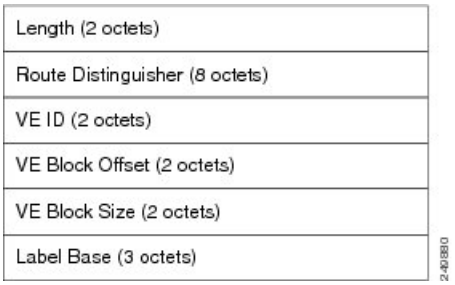
- A means for a PE to learn which remote PEs are members of a given VPLS. This process is known as autodiscovery.
- A means for a PE to learn the pseudowire label expected by a given remote PE for a given VPLS. This process is known as signaling.

The BGP Network Layer Reachability Information (NLRI) takes care of the above two components simultaneously. The NLRI generated by a given PE contains the necessary information required by any other PE. These components enable the automatic setting up of a full mesh of pseudowires for each VPLS without having to manually configure those pseudowires on each PE.

NLRI Format for VPLS with BGP AD and Signaling

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

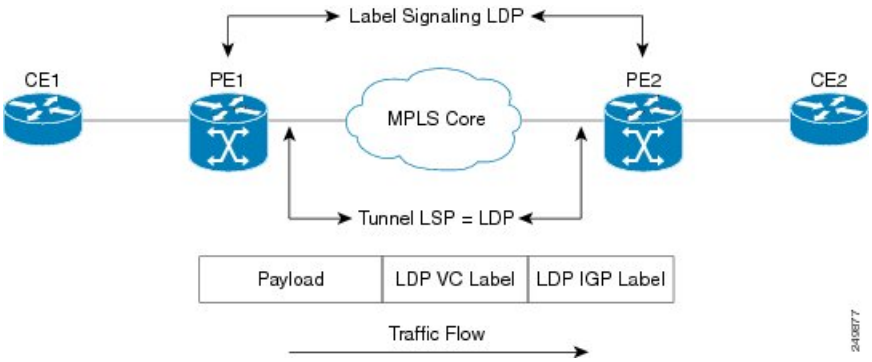
Figure 5: NLRI Format



BGP Auto Discovery With LDP Signaling

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

Figure 6: Discovery and Signaling Attributes



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

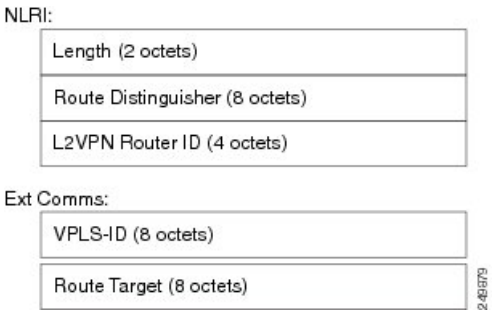
The LDP protocol is used to configure a pseudowire to all the other PE routers. FEC 129 is used for the signaling. The information carried by FEC 129 includes the VPLS ID, the Target Attachment Individual Identifier (TAII) and the Source Attachment Individual Identifier (SAII).

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

NLRI and Extended Communities

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

Figure 7: NLRI and Extended Communities



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 maintained on the route processor (RP) card.

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

MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To 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 propagated and programs the hardware for the router.

Note Static MAC move is not supported from one port, interface, or AC to another port, interface, or AC. For example, if a static MAC is configured on AC1 (port 1) and then, if you send a packet with the same MAC as source MAC on AC2 (port 2), then you can't attach this MAC to AC2 as a dynamic MAC. Therefore, do not send any packet with a MAC as any of the static MAC addresses configured.

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.

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

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

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 these 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. 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 for LDP signalled FEC128 pseudowires is limited to manually configured VPLS pseudowires. In addition, Cisco IOS XR software supports LSP ping for point-to-point single-segment pseudowires that are signalled using LDP FEC129 AII-type 2 applicable to VPWS or signalled using LDP FEC129 AII-type 1 applicable to VPLS. For information about Virtual Circuit Connection Verification (VCCV) support and the **ping mpls pseudowire** command, see the *MPLS Command Reference for the Cisco CRS Router*.

VPLS Scalability and Performance Targets

The Cisco CRS-1 router employs the ternary content addressable memory (TCAM) to meet the performance and scalable targets over VPLS.

The following table below describes the scalability and performance targets for the Cisco CRS-1 router.

Table 1: VPLS Scalability and Performance Targets

Performance	Scalability Target
Maximum bridge domains per Line Card	1024
Maximum bridge domains per system	1024
Maximum MACs per bridge domain	15999
Maximum MACs per Line Card	65536
Maximum MACs per system	65536
Maximum attachment circuits per bridge domain	4085
Maximum pseudowires per bridge domain	256
Maximum pseudowires per system	16340

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.

It is recommended to enable pseudowire status time length value (TLV) for optimal switchover performance.

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.

Pseudowire Headend

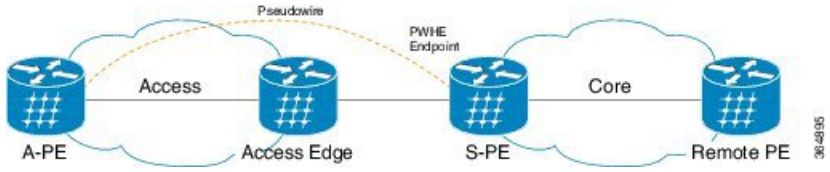
Pseudowires (PWs) enable payloads to be transparently carried across IP/MPLS packet-switched networks (PSNs). PWs are regarded as simple and manageable lightweight tunnels for returning customer traffic into core networks. Service providers are now extending PW connectivity into the access and aggregation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

Pseudowire Headend (PWHE) is a technology that allows termination of access pseudowires (PWs) into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWs provide an easy and scalable mechanism for

tunneling customer traffic into a common IP/MPLS network infrastructure. PWHE allows customers to provision features such as QOS access lists (ACL), L3VPN on a per PWHE interface basis, on a service Provider Edge (PE) router.

PWHE cross-connects to a pseudowire neighbour, which is reachable through recursive as well as non-recursive prefix. The reachability through recursive prefix is through introduction of BGP RFC3107 support on the Cisco CRS Router. Consider the following network topology for an example scenario.

Figure 8: Pseudowire Network



For PWHE x-connect configuration, interconnectivity between A-PE (Access Provider Edge) and S-PE is through BGP RFC3107 that distributes MPLS labels along with IP prefixes. The customer network can avoid using an IGP to provide connectivity to the S-PE device, which is outside the customer's autonomous system.

For all practical purposes, the PWHE interface is treated like any other existing L3 interface. PWs operate in one of the following modes:

- Bridged interworking (VC type 5 or VC type 4)
- IP interworking mode (VC type 11)

With VC type 4 and VC type 5, PWs carry customer Ethernet frames (tagged or untagged) with IP payload. Thus, an S-PE device must perform ARP resolution for customer IP addresses learned over the PWHE. With VC type 4 (VLAN tagged) and VC type 5 (Ethernet port/raw), PWHE acts as a broadcast interface. Whereas with VC type 11 (IP Interworking), PWHE acts as a point-to-point interface. Therefore there are two types of PWHE interface—PW-Ether (for VC type 4 and 5) and PW-IW (for VC type 11). These PWs can terminate into a VRF or the IP global table on S-PE.

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.

Pseudowire Grouping

When pseudowires (PWs) are established, each PW is assigned a group ID that is common for all PWs created on the same physical port. When a physical port becomes non-functional or disabled, Automatic Protection Switching (APS) signals the peer router to get activated and L2VPN sends a single message to advertise the status change of all PWs that have the Group ID associated with the physical port. A single L2VPN signal thus avoids a lot of processing and loss in reactivity.

Note Pseudowire grouping is disabled by default.

How to Implement Multipoint Layer 2 Services

This section describes the tasks that are required to implement Multipoint Layer 2 Services:

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#
```

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

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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.

Step 6 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Exits the current configuration mode.

Step 7 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

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

Enabling Pseudowire Status TLV

When a pseudowire is setup, label distribution protocol (LDP) determines the method for signaling pseudowire status. Cisco IOS-XR provides a configuration option that allows you to enable pseudowire status type length value (TLV).

Note Unless pseudowire status TLV is explicitly enabled under L2VPN configuration, the default signaling method is Label Withdrawal. Pseudowire status TLV must be enabled on both local and remote PEs. If only one provider edge router is configured with the **pw-status tlv** command, then label withdrawal method is used.

Perform this task to enable pseudowire status TLV.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-status tlv**
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP00/CPU0:router(config)# l2vpn
RP/0/RP00/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **pw-status tlv**

Example:

```
RP/0/RP00/CPU0:router(config-l2vpn)# pw-status tlv
```

Enables pseudowire status TLV.

Step 4 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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** *value*
6. **backup neighbor** *ip-address* **pw-id** *number*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **xconnect group** *group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group A
RP/0/RP0/CPU0:router(config-l2vpn-xc)#
```

Enters the name of the cross-connect group.

Step 4 **p2p** *xconnect-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)#
```

Enters a name for the point-to-point cross-connect.

Step 5 **neighbor** *ip-address* **pw-id** *value*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire segment for the cross-connect.

Step 6 **backup neighbor** *ip-address* **pw-id** *number*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# backup neighbor 1.1.1.1 pw-id 2
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#
```

Configures the backup pseudowire for the cross-connect.

Step 7 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **pw-class** *class-name*

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/RP0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-pwc)# exit
```

Exits the pseudowire class submode.

Step 6 **xconnect group** *group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group A
RP/0/RP0/CPU0:router(config-l2vpn-xc)#
```

Enters the name of the cross-connect group.

Step 7 **p2p** *xconnect name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)#
```

Enters a name for the point-to-point cross-connect.

Step 8 **neighbor** *ip-address* **pw-id** *number*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire segment for the cross-connect.

Step 9 **pw-class** *class-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class class_1
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire class name.

Step 10 **backup neighbor** *ip-address* **pw-id** *number*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 1.1.1.1 pw-id 2
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#
```

Configures the backup pseudowire for the point-to-point neighbor.

Step 11 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Associating Members with a Bridge Domain

After a bridge domain is created, perform this task to assign interfaces to the bridge domain. These 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. *(Optional)* **static-mac-address** { *MAC-address* }
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0
RP/0/RP0/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.

Step 6 *(Optional)* **static-mac-address** { *MAC-address* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# static-mac-address 1.1.1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

Step 7 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Bridge Domain Parameters

To configure bridge domain parameters, associate these 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—Flooding is enabled always.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the l2vpn configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# flooding disable
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Disables flooding.

Step 6 **mtu** *bytes*

Example:

```
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

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

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
```

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5 **shutdown**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# shutdown
```

Shuts down a bridge domain to bring the bridge and all attachment circuits and pseudowires under it to admin down state.

Step 6 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a Layer 2 Virtual Forwarding Instance

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

Creating the Virtual Forwarding Instance

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
```

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a 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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a 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 **static-mac-address** { *MAC-address* }

Example:

```
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

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

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
```

```
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a 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 **pw-class** { *class-name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada
```

Configures the pseudowire class template name to use for the pseudowire.

Step 8 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Pseudowires 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a 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 **mpls static label** { **local** *value* } { **remote** *value* }

Example:

```
RP/0/RP0/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 pseudowire configuration. You can set the local and remote pseudowire labels.

Step 8 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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. Use the **commit** or **end** command.
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown
```

Disables the virtual forwarding interface (VFI).

Step 7 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 8 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/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:

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-domainname*
5. **mac**
6. **learning disable**
7. Use the **commit** or **end** command.
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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-domainname*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **learning disable**

Example:

```
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 8 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/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. Use the **commit** or **end** command.
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **withdraw { access-pw disable | disable }**

Example:

```
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 8 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/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. *(Optional)* **interface** *type interface_id*
6. **mac**
7. **limit**
8. **maximum** { *value* }

9. **action** { **flood** | **no-flood** | **shutdown** }
10. **notification** { **both** | **none** | **trap** }
11. **mac limit threshold** *80*
12. Use the **commit** or **end** command.
13. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 *(Optional)* **interface** *type interface_id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface gigabitEthernet 0/2/0/1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters the interface configuration mode of the specified interface and adds this interface as the bridge domain member interface.

Note Run this step if you want to configure the MAC address limit only for a specific interface. The further steps show the router prompt displayed when you have skipped this step to configure the MAC address limit at the bridge domain level.

Step 6 **mac**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 7 **limit**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# limit
RP/0/RP0/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 8 **maximum** { *value* }

Example:

```
RP/0/RP0/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.

Step 9 **action** { **flood** | **no-flood** | **shutdown** }

Example:

```
RP/0/RP0/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 10 **notification** { **both** | **none** | **trap** }

Example:

```
RP/0/RP0/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 11 **mac limit threshold** *80*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# mac limit threshold 80
```

Configures the MAC limit threshold. The default is 75% of MAC address limit configured in step 8.

Step 12 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 13 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/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. Use the **commit** or **end** command.
9. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

```
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **aging**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# aging
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#
```

Enters the MAC aging configuration submode to set the aging parameters such as time and type.

Step 7 **time** { *seconds* }

Example:

```
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Step 9 **show l2vpn bridge-domain** [**detail**]

Example:

```
RP/0/RP0/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 or bridge port level. 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6 **port-down flush disable**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#
port-down flush disable
```

Disables MAC flush when the bridge port becomes nonfunctional.

Step 7 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring VPLS with BGP Autodiscovery and Signaling

Perform this task to configure BGP-based autodiscovery and signaling.

SUMMARY STEPS

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

7. **autodiscovery bgp**
8. **rd** { *as-number:nn* | *ip-address:nn* | **auto** }
9. **route-target** { *as-number:nn* | *ip-address:nn* | **export** | **import** }
10. **route-target import** { *as-number:nn* | *ip-address:nn* }
11. **route-target export** { *as-number:nn* | *ip-address:nn* }
12. **signaling-protocol bgp**
13. **ve-id** { *number*}
14. **ve-range** { *number* }
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group metroA
```

Enters configuration mode for the named bridge group.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain east
```

Enters configuration mode for the named bridge domain.

Step 5 **vfi** { *vfi-name* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east
```

Enters virtual forwarding instance (VFI) configuration mode.

Step 6 **vpn-id** *vpn-id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# vpn-id 100
```

Specifies the identifier for the VPLS service. The VPN ID has to be globally unique within a PE router. i.e., the same VPN ID cannot exist in multiple VFIs on the same PE router. In addition, a VFI can have only one VPN ID.

Step 7 **autodiscovery bgp**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
```

Enters BGP autodiscovery configuration mode where all BGP autodiscovery parameters are configured.

This command is not provisioned to BGP until at least the VPN ID and the signaling protocol is configured.

Step 8 **rd** { *as-number:nn* | *ip-address:nn* | **auto** }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# rd auto
```

Specifies the route distinguisher (RD) under the VFI.

The RD is used in the BGP NLRI to identify VFI. Only one RD can be configured per VFI, and except for **rd auto** the same RD cannot be configured in multiple VFIs on the same PE.

When **rd auto** is configured, the RD value is as follows: {BGP Router ID}:{16 bits auto-generated unique index}.

Step 9 **route-target** { *as-number:nn* | *ip-address:nn* | **export** | **import** }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target 500:99
```

Specifies the route target (RT) for the VFI.

At least one import and one export route targets (or just one route target with both roles) need to be configured in each PE in order to establish BGP autodiscovery between PEs.

If no export or import keyword is specified, it means that the RT is both import and export. A VFI can have multiple export or import RTs. However, the same RT is not allowed in multiple VFIs in the same PE.

Step 10 **route-target import** { *as-number:nn* | *ip-address:nn* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target import 200:20
```

Specifies the import route target for the VFI.

The PE compares import route target with the RT in the received NLRI: the RT in the received NLRI must match the import RT to determine that the RTs belong to the same VPLS service.

Step 11 **route-target export** { *as-number:nn* | *ip-address:nn* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target export 100:10
```

Specifies the export route target for the VFI.

Export route target is the RT that is going to be in the NLRI advertised to other PEs.

Step 12 **signaling-protocol bgp**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# signaling-protocol bgp
```

Enables BGP signaling, and enters the BGP signaling configuration submode where BGP signaling parameters are configured.

This command is not provisioned to BGP until VE ID and VE ID range is configured.

Step 13 **ve-id** { *number*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# ve-id 10
```

Specifies the local PE identifier for the VFI for VPLS configuration.

The VE ID identifies a VFI within a VPLS service. This means that VFIs in the same VPLS service cannot share the same VE ID. The scope of the VE ID is only within a bridge domain. Therefore, VFIs in different bridge domains within a PE can use the same VE ID.

Step 14 **ve-range** { *number* }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# ve-range 40
```

Overrides the minimum size of VPLS edge (VE) blocks.

The default minimum size is 10. Any configured VE range must be higher than 10.

Step 15 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring VPLS with BGP Autodiscovery and LDP Signaling

Perform this task to configure BGP-based Autodiscovery and signaling:

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **router-id** *ip-address*
4. **bridge group** *bridge-group-name*
5. **bridge-domain** *bridge-domain-name*
6. **vfi** {*vfi-name*}
7. **autodiscovery bgp**
8. **vpn-id** *vpn-id*
9. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}
10. **route-target** {*as-number:nn* | *ip-address:nn* | **export** | **import** }
11. **route-target import** {*as-number:nn* | *ip-address:nn*}
12. **route-target export** {*as-number:nn* | *ip-address:nn*}
13. **signaling-protocol ldp**
14. **vpls-id** {*as-number:nn* | *ip-address:nn*}
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **router-id** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# router-id 1.1.1.1
```

Specifies a unique Layer 2 (L2) router ID for the provider edge (PE) router.

The router ID must be configured for LDP signaling, and is used as the L2 router ID in the BGP NLRI, SAII (local L2 Router ID) and TAII (remote L2 Router ID). Any arbitrary value in the IPv4 address format is acceptable.

Note Each PE must have a unique L2 router ID. This CLI is optional, as a PE automatically generates a L2 router ID using the LDP router ID.

Step 4 **bridge group** *bridge-group-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group metroA
```

Enters configuration mode for the named bridge group.

Step 5 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain east
```

Enters configuration mode for the named bridge domain.

Step 6 **vfi** {*vfi-name*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east
```

Enters virtual forwarding instance (VFI) configuration mode.

Step 7 **autodiscovery bgp**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
```

Enters BGP autodiscovery configuration mode where all BGP autodiscovery parameters are configured.

This command is not provisioned to BGP until at least the VPN ID and the signaling protocol is configured.

Step 8 **vpn-id** *vpn-id*

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# vpn-id 100
```

Specifies the identifier for the VPLS service. The VPN ID has to be globally unique within a PE router. i.e., the same VPN ID cannot exist in multiple VFIs on the same PE router. In addition, a VFI can have only one VPN ID.

Step 9 **rd** {*as-number:nn* | *ip-address:nn* | **auto**}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# rd auto
```

Specifies the route distinguisher (RD) under the VFI.

The RD is used in the BGP NLRI to identify VFI. Only one RD can be configured per VFI, and except for **rd auto** the same RD cannot be configured in multiple VFIs on the same PE.

When **rd auto**is configured, the RD value is as follows: {BGP Router ID}:{16 bits auto-generated unique index}.

Step 10 **route-target** {*as-number:nn* | *ip-address:nn* | **export** | **import** }

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target 500:99
```

Specifies the route target (RT) for the VFI.

At least one import and one export route targets (or just one route target with both roles) need to be configured in each PE in order to establish BGP autodiscovery between PEs.

If no export or import keyword is specified, it means that the RT is both import and export. A VFI can have multiple export or import RTs. However, the same RT is not allowed in multiple VFIs in the same PE.

Step 11 **route-target import** {*as-number:nn* | *ip-address:nn*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target import 200:20
```

Specifies the import route target for the VFI.

The PE compares the import route target with the RT in the received NLRI: the RT in the received NLRI must match the import RT to determine that the RTs belong to the same VPLS service.

Step 12 **route-target export** {*as-number:nn* | *ip-address:nn*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target export 100:10
```

Specifies the export route target for the VFI.

Export route target is the RT that is going to be in the NLRI advertised to other PEs.

Step 13 **signaling-protocol ldp**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# signaling-protocol ldp
```

Enables LDP signaling.

Step 14 **vpls-id** {*as-number:nn* | *ip-address:nn*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# vpls-id 10:20
```

Specifies VPLS ID which identifies the VPLS domain during signaling.

This command is optional in all PEs that are in the same Autonomous System (share the same ASN) because a default VPLS ID is automatically generated using BGP's ASN and the configured VPN ID (i.e., the default VPLS ID equals ASN:VPN-ID). If an ASN of 4 bytes is used, the lower two bytes of the ASN are used to build the VPLS ID. In case of InterAS, the VPLS ID must be explicitly configured. Only one VPLS ID can be configured per VFI, and the same VPLS ID cannot be used for multiple VFIs.

Step 15 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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 configuration restrictions are applicable for PWHE:

- Up to 4096 PWHE interfaces (a combination of pw-ether and pw-iw).
- Up to eight interface lists per peer.
- Up to eight 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 be configured on CRS only.
- Interface lists can accept POS, GigabitEthernet, TenGigabitEthernet, SRP, Bundle Ethernet and Bundle POS; 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, that is, to attach the generic interface list with a PWHE interfaces.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface pw-ether** *id*

Example:

```
RP/0/RP0/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/RP0/CPU0:router(config-if)# attach generic-interface-list interfacelist1
```

Attaches the generic interface list to the PW-Ether or PW-IW interface . To remove the generic interface list from the PW-Ether or PW-IW interface, use the **no** form of the command, that is, **no generic-interface-list** *list-name*

Step 4 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface pw-ether** *id*

Example:

```
RP/0/RP0/CPU0:router(config)# interface pw-ether <id>
or
RP/0/RP0/CPU0:router(config)# interface pw-iw <id>
```

(**interface pw-ether** *id*) Configures the PWHE interface and enters the interface configuration mode.

(**interface pw-iw** *id*) Configures the PWHE interface and enters the interface configuration mode.

Step 3 **attach generic-interface-list** *interface_list_name*

Example:

```
RP/0/RP0/CPU0:router(config-if)# attach generic-interface-list interfacelist1
```

Attaches the generic interface list to the PW-Ether or PW-IW interface.

Step 4 **l2overhead** *bytes*

Example:

```
RP/0/RP0/CPU0:router(config-if)# l2overhead 20
```

Sets layer 2 overhead size.

Step 5 **load-interval** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-if)# load-interval 90
```

Specifies interval, in seconds, for load calculation for an interface.

The interval:

- Can be set to 0 [0 disables load calculation]
- If not 0, must be specified in multiples of 30 between 30 and 600.

Step 6 **dampening** *decay-life*

Example:

```
RP/0/RP0/CPU0:router(config-if)# dampening 10
```

Configures state dampening on the given interface (in minutes).

Step 7 **logging events link-status**

Example:

```
RP/0/RP0/CPU0:router(config-if)# logging events link-status
```

Configures per interface logging.

Step 8 **mac-address** *MAC address*

Example:

```
RP/0/RP0/CPU0:router(config-if)# mac-address aaaa.bbbb.cccc
```

Sets the MAC address (xxxx.xxxx.xxxx) on an interface.

Step 9 **mtu** *interface_MTU*

Example:

```
RP/0/RP0/CPU0:router(config-if)# mtu 128
```

Sets the MTU on an interface.

Step 10 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 3 **xconnect group** *group-name*

Example:

```
RP/0/RP0/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/RP0/CPU0:router (config-l2vpn-xc)# p2p ms-pw1
```

Enters P2P configuration submode.

Step 5 **interface pw-ether** *id*

Example:

```
RP/0/RP0/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/RP0/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.

Step 7 **pw-class** *class-name*

Example:

```
RP/0/RP0/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 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Enabling Pseudowire Grouping

Perform this task to enable pseudowire grouping.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **pw-grouping**

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# pw-grouping
```

Enables pseudowire grouping

Step 4 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Examples for Multipoint Layer 2 Services

This section includes these configuration examples:

Multipoint Layer 2 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 Multipoint Layer 2 Services provider edge (PE) nodes.

This configuration example shows how to configure PE 1:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE1-VPLS-A
    interface TenGigE0/0/0/0
```

```
    vfi 1
     neighbor 10.2.2.2 pw-id 1
     neighbor 10.3.3.3 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 10.1.1.1 255.255.255.255
```

This configuration example shows how to configure PE 2:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE2-VPLS-A
    interface TenGigE0/0/0/1

    vfi 1
     neighbor 10.1.1.1 pw-id 1
     neighbor 10.3.3.3 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 10.2.2.2 255.255.255.255
```

This configuration example shows how to configure PE 3:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE3-VPLS-A
    interface TenGigE0/0/0/2
    vfi 1
     neighbor 10.1.1.1 pw-id 1
     neighbor 10.2.2.2 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 10.3.3.3 255.255.255.255
```

Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example

This configuration shows how to configure Multipoint Layer 2 Services for a PE-to-CE nodes:

```
configure
 interface TenGigE0/0/0/0
  l2transport---AC interface
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end
```

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
```

Disabling MAC Flush: Examples

You can disable the MAC flush at these 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:

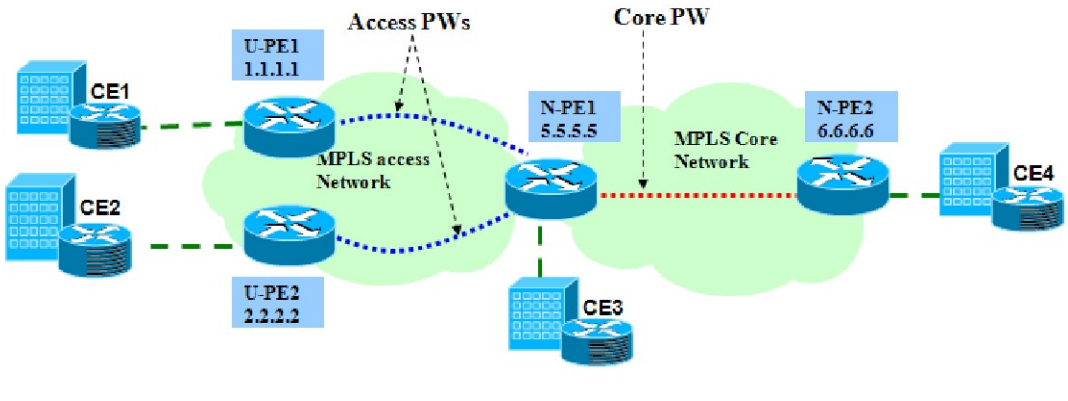
```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    mac
    port-down flush disable
  end
```

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

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

H-VPLS Configuration: Examples

This example shows how to configure hierarchical VPLS (H-VPLS). All examples in this section are based on the following topology where N-PE1 is the H-VPLS Node:



VPLS with QinQ or QinAny: Example

Global Interface Configuration at N-PE1:

```
interface GigabitEthernet0/0/0/0
dot1q tunneling ethertype 0x9200
!
interface GigabitEthernet0/0/0/1
dot1q tunneling ethertype 0x9100
!
interface GigabitEthernet0/0/0/0.1 l2transport
encapsulation dot1q 20 second-dot1q 21
!
interface GigabitEthernet0/0/0/1.1 l2transport
encapsulation dot1q 10 second-dot1q any
```

L2VPN Configuration at N-PE1:

```
l2vpn
 bridge group g1
  bridge-domain d1
   interface GigabitEthernet0/0/0/0.1
   !
   interface GigabitEthernet0/0/0/1.1
   !
  vfi core-pws
     neighbor 6.6.6.6 pw-id 10
```

Global Interface Configuration at N-PE2:

```
interface GigabitEthernet0/6/0/0
dot1q tunneling ethertype 0x9200
 !
interface GigabitEthernet0/6/0/1
dot1q tunneling ethertype 0x9100
```

```
!
interface GigabitEthernet0/6/0/0.1 l2transport
encapsulation dot1q 10 second-dot1q 20
!
interface GigabitEthernet0/6/0/1.1 l2transport
encapsulation dot1q 1 second-dot1q 2
```

L2VPN Configuration at N-PE2:

```
l2vpn
 bridge group g1
  bridge-domain d1
   interface GigabitEthernet0/6/0/0.1
   !
   interface GigabitEthernet0/6/0/1.1
   !
   vfi core-pws
    neighbor 5.5.5.5 pw-id 10
```

H-VPLS with Access-PWs: Example

Router Configuration at U-PE1:

```
l2vpn
 pw-class vpls
  encapsulation mpls
    transport-mode ethernet
!
 xconnect group g1
  p2p p1
   interface GigabitEthernet0/1/1/0.1 --> Local AC
   neighbor 5.5.5.5 pw-id 100 --> Access PW to N-PE1
    pw-class vpls
interface GigabitEthernet0/1/1/0.1 l2transport
encapsulation dot1q 1
```

Router Configuration at U-PE2:

```
l2vpn
 pw-class vpls
  encapsulation mpls
   transport-mode ethernet
    mac-withdraw
!
xconnect group g1
  p2p p1
   interface GigabitEthernet0/2/5/0.1 --> Local AC
   neighbor 5.5.5.5 pw-id 100 --> Access PW to N-PE1
    pw-class vpls
interface GigabitEthernet0/2/5/0.1 l2transport
encapsulation dot1q 1
```

Router Configuration at N-PE1:

```
l2vpn
 bridge group g1
  bridge-domain d1
   interface GigabitEthernet0/1/4/0.1 ? Local AC
```

```
   neighbor 1.1.1.1 pw-id 100 --> Access PW to U-PE1
   neighbor 2.2.2.2 pw-id 100 --> Access PW to U-PE2
!
   vfi core1
     neighbor 6.6.6.6 pw-id 100 --> Core PW to N-PE2
interface GigabitEthernet0/1/4/0.1 l2transport
encapsulation dot1q 1
```

Router Configuration at N-PE2:

```
l2vpn
 bridge group g1
  bridge-domain d1
   interface GigabitEthernet0/2/1/0.1 --> Local AC
   vfi core1
    neighbor 5.5.5.5 pw-id 100 --> Core PW to N-PE1
interface GigabitEthernet0/2/1/0.1 l2transport
  encapsulation dot1q 1
```

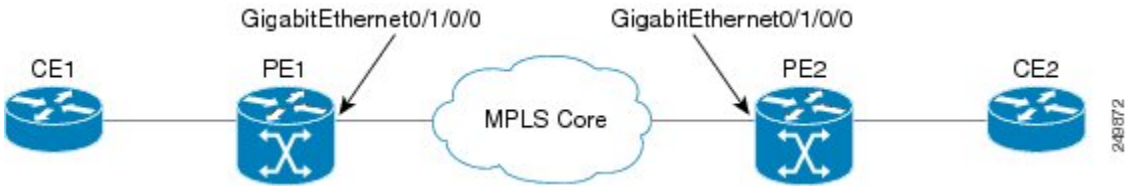
Configuring VPLS with BGP Autodiscovery and Signaling: Example

This section contains these configuration examples for configuring the BGP autodiscovery and signaling feature:

LDP and BGP Configuration

The following figure illustrates an example of LDP and BGP configuration.

Figure 9: LDP and BGP Configuration



Configuration at PE1:

```
interface Loopback0
 ipv4 address 1.1.1.100 255.255.255.255
!
interface Loopback1
 ipv4 address 1.1.1.10 255.255.255.255
!
mpls ldp
 router-id 1.1.1.1
 interface GigabitEthernt0/1/0/0
!
router bgp 120
 address-family l2vpn vpls-vpws
!
 neighbor 2.2.2.20
 remote-as 120
 update-source Loopback1
 address-family l2vpn vpls-vpws
 signaling bgp disable
```

Configuration at PE2:

```
interface Loopback0
 ipv4 address 2.2.2.200 255.255.255.255
!
interface Loopback1
 ipv4 address 2.2.2.20 255.255.255.255
!
mpls ldp
 router-id 2.2.2.2
 interface GigabitEthernt0/1/0/0
!
router bgp 120
 address-family l2vpn vpls-vpws
!
 neighbor 1.1.1.10
 remote-as 120
 update-source Loopback1
 address-family l2vpn vpls-vpws
```

Minimum L2VPN Configuration for BGP Autodiscovery with BGP Signaling

This example illustrates the minimum L2VPN configuration required for BGP Autodiscovery with BGP Signaling, where any parameter that has a default value is not configured.

```
(config)# l2vpn
(config-l2vpn)# bridge group {bridge group name}
(config-l2vpn-bg)# bridge-domain {bridge domain name}
(config-l2vpn-bg-bd)# vfi {vfi name}
(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
(config-l2vpn-bg-bd-vfi-ad)# vpn-id 10
(config-l2vpn-bg-bd-vfi-ad)# rd auto
(config-l2vpn-bg-bd-vfi-ad)# route-target 1.1.1.1:100
(config-l2vpn-bg-bd-vfi-ad-sig)# signaling-protocol bgp
(config-l2vpn-bg-bd-vfi-ad-sig)# ve-id 1
(config-l2vpn-bg-bd-vfi-ad-sig)# commit
```

VPLS with BGP Autodiscovery and BGP Signaling

The following figure illustrates an example of configuring VPLS with BGP autodiscovery (AD) and BGP Signaling.

Figure 10: VPLS with BGP autodiscovery and BGP signaling

Configuration at PE1:

```
l2vpn
  bridge group gr1
   bridge-domain bd1
    interface GigabitEthernet0/1/0/1.1
    vfi vf1
    ! AD independent VFI attributes
```

```
    vpn-id 100
    ! Auto-discovery attributes
    autodiscovery bgp
    rd auto
    route-target 2.2.2.2:100
    ! Signaling attributes
    signaling-protocol bgp
    ve-id 3
```

Configuration at PE2:

```
l2vpn
  bridge group gr1
   bridge-domain bd1
    interface GigabitEthernet0/1/0/2.1
    vfi vf1
    ! AD independent VFI attributes
    vpn-id 100
    ! Auto-discovery attributes
    autodiscovery bgp
    rd auto
    route-target 2.2.2.2:100
    ! Signaling attributes
    signaling-protocol bgp
    ve-id 5
```

This is an example of NLRI for VPLS with BGP AD and signaling:

Discovery Attributes

NLRI sent at PE1:

```
Length = 19
Router Distinguisher = 3.3.3.3:32770
VE ID = 3
VE Block Offset = 1
VE Block Size = 10
Label Base = 16015
```

NLRI sent at PE2:

```
Length = 19
Router Distinguisher = 1.1.1.1:32775
VE ID = 5
VE Block Offset = 1
VE Block Size = 10
Label Base = 16120
```

Minimum Configuration for BGP Autodiscovery with LDP Signaling

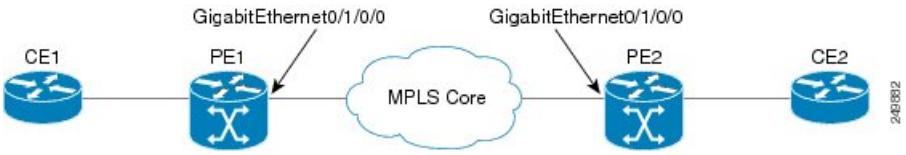
This example illustrates the minimum L2VPN configuration required for BGP Autodiscovery with LDP Signaling, where any parameter that has a default value is not configured.

```
(config)# l2vpn
(config-l2vpn)# bridge group {bridge group name}
(config-l2vpn-bg)# bridge-domain {bridge domain name}
(config-l2vpn-bg-bd)# vfi {vfi name}
(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
(config-l2vpn-bg-bd-vfi-ad)# vpn-id 10
(config-l2vpn-bg-bd-vfi-ad)# rd auto
(config-l2vpn-bg-bd-vfi-ad)# route-target 1.1.1.1:100
(config-l2vpn-bg-bd-vfi-ad)# commit
```

VPLS with BGP Autodiscovery and LDP Signaling

The following figure illustrates an example of configuring VPLS with BGP autodiscovery (AD) and LDP Signaling.

Figure 11: VPLS with BGP autodiscovery and LDP signaling



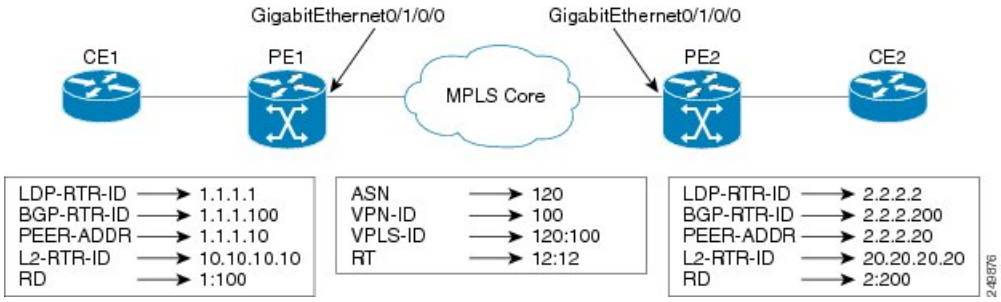
Configuration at PE1:

```
l2vpn
  router-id 10.10.10.10
  bridge group bg1
   bridge-domain bd1
    vfi vf1
    vpn-id 100
    autodiscovery bgp
    rd 1:100
    router-target 12:12
```

Configuration at PE2:

```
l2vpn
  router-id 20.20.20.20
  bridge group bg1
   bridge-domain bd1
    vfi vf1
     vpn-id 100
     autodiscovery bgp
      rd 2:200
      router-target 12:12
      signaling-protocol ldp
       vpls-id 120:100
```

Discovery and Signaling Attributes



Configuration at PE1:

```
LDP Router ID - 1.1.1.1
BGP Router ID - 1.1.1.100
Peer Address - 1.1.1.10
L2VPN Router ID - 10.10.10.10
Route Distinguisher - 1:100
```

Common Configuration between PE1 and PE2:

```
ASN - 120
VPN ID - 100
VPLS ID - 120:100
Route Target - 12:12
```

Configuration at PE2:

```
LDP Router ID - 2.2.2.2
BGP Router ID - 2.2.2.200
Peer Address - 2.2.2.20
L2VPN Router ID - 20.20.20.20
Route Distinguisher - 2:200
```

Discovery Attributes

NLRI sent at PE1:

```
Source Address - 1.1.1.10
Destination Address - 2.2.2.20
Length - 14
Route Distinguisher - 1:100
L2VPN Router ID - 10.10.10.10
VPLS ID - 120:100
Route Target - 12:12
```

NLRI sent at PE2:

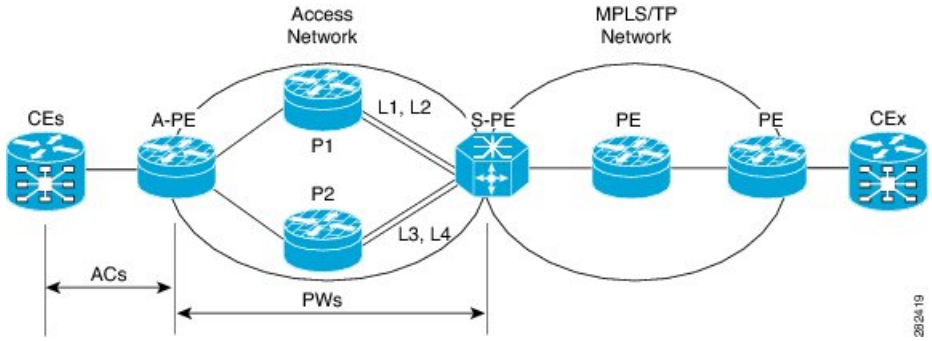
```
Source Address - 2.2.2.20
Destination Address - 1.1.1.10
Length - 14
Route Distinguisher - 2:200
L2VPN Router ID - 20.20.20.20
```

```
VPLS ID - 120:100
Route Target - 12:12
```

Configuring Pseudowire Headend: Example

This section provides an example of pseudowire headend configuration.

Figure 12: 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).

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
```

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.

P1

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

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
```

Enabling Pseudowire Grouping: Example

This example shows how to enable pseudowire grouping.

```
config
l2vpn
 pw-grouping
```