# Configure 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.

# Prerequisites for Implementing Multipoint Layer 2 Services

Before configuring Multipoint Layer 2 Services, ensure that these tasks and conditions are met:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

  If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- Configure IP routing in the core so that the provider edge (PE) routers can reach each other through IP.

- 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.

# Information About Implementing Multipoint Layer 2 Services

To implement Multipoint Layer 2 Services, you must 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**　VPLS PW is not supported over BGP multipath.

Some of the components present in a Multipoint Layer 2 Services network are described in these sections.

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

## 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.

## Pseudowires

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

## Access Pseudowire is not supported over VPLS Bridge Domain

Access PW is not supported over VPLS bridge domain. Only core PW which is configured under VFI is supported.

Configuration Example

```
l2vpn
 bridge group bg1
  bridge-domain l2vpn
```

```
   interface TenGigE0/0/0/13.100
   !
 vfi 1
  neighbor 192.0.2.1 pw-id 12345
   pw-class mpls_csr
  !
 !
!
```

## Virtual Forwarding Instance

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

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

# 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. A bridge domain, which is the building block for multipoint bridging, is present on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment circuits. The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

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 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. This also requires all of the PE routers that are participating in a VPLS instance to form emulated virtual circuits (VCs) among them.

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

# VPLS for Layer 2 Switching

VPLS technology includes the capability of configuring the router to perform Layer 2 bridging. In this mode, the router can be configured to operate like other Cisco switches.

**Note** The storm control configuration is supported only on one sub-interface under a main interface, though the system allows you to configure storm control on more than one sub-interface. However, only the first storm control configuration under a main interface takes effect, though the running configuration shows all the storm control configurations that are committed. After reload, any of the storm control configurations may take effect irrespective of the order of configuration.

These features are supported:

- Bridging IOS XR Trunk Interfaces

- Bridging on EFPs

# 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
```

# 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.

## 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. The default value for the MAC address limit is 64000 for Cisco NCS 5501 and Cisco NCS 5502.

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)

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

MAC address limit action applies only when the number of local MAC addresses exceeds the configured limit. The software unlearns the MAC addresses until it reaches the configured MAC limit threshold value.

Later, the router restarts learning new MAC addresses. In the event when the MAC limit threshold is not configured, the default threshold is 75% of the configured MAC address limit.

## MAC Address Withdrawal

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

For the Cisco IOS XR VPLS implementation, a portion of the dynamically learned MAC addresses are cleared by using the MAC addresses aging mechanism by default. The MAC address withdrawal feature is added through the LDP Address Withdrawal message. To enable the MAC address withdrawal feature, use the **withdrawal** command in l2vpn bridge group bridge domain MAC configuration mode. To verify that the MAC address withdrawal is enabled, use the **show l2vpn bridge-domain** command with the **detail** keyword.

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

The LDP MAC Withdrawal feature is generated due to 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.

# How to Implement 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 .

**Procedure**

**Step 1**  **configure**

**Example:**

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

Enters the XR Config 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.

## 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

**Procedure**

**Step 1**    **configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1**     **configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1**      **configure**

**Example:**

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

Enters the XR Config 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)#
```

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.

**Procedure**

**Step 1**  **configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1** **configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1**　　**configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1**     **configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1** **configure**

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

Enters the XR Config 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.

**Procedure**

**Step 1**   **configure**

**Example:**

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

Enters the XR Config 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 cisco
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.

**Note**   The **show l2vpn forwarding bridge-domain BRIDGE_GROUP:BRIDGE_DOMAIN mac-address location R/S/I** command does not automatically dump MAC address hardware information. The show output information might not be current. Perform any of the following actions before executing the **show l2vpn forwarding bridge-domain BRIDGE_GROUP:BRIDGE_DOMAIN mac-address location R/S/I** command:

- Resynchronize the MAC address entries by executing **l2vpn resynchronize forwarding mac-address location R/S/I** command.

- Dump the MAC address table by running **show l2vpn forwarding bridge-domain mac-address location R/S/I** command.

# Configuring the MAC Address Source-based Learning

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

**Procedure**

**Step 1**  **configure**

**Example:**

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

Enters the XR Config 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
```

**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.

# Configuring the MAC Address Limit

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

**Procedure**

**Step 1**     **configure**

**Example:**

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

Enters the XR Config 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.

**Procedure**

**Step 1**    **configure**

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

Enters the XR Config 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 cisco
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. 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.

**Procedure**

**Step 1** **configure**

**Example:**
```
RP/0/RP0/CPU0:router# configure
```
Enters the XR Config 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.

# 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 10.1.1.1 255.255.255.25
```

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 10.2.2.2 255.255.255.25
```

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 10.3.3.3 255.255.255.25
```

# 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

   no ipv4 address
   no ipv4 directed-broadcast
   negotiation auto
   no cdp enable
```

# Displaying MAC Address Withdrawal Fields: Example

This sample output shows the MAC address withdrawal fields:

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain detail

Legend: pp = Partially Programmed.
Bridge group: 222, bridge-domain: 222, id: 0, state: up, ShgId: 0, MSTi: 0
```

```
   Coupled state: disabled
MAC learning: enabled
MAC withdraw: enabled
  MAC withdraw sent on: bridge port up
  MAC withdraw relaying (access to access): disabled
Flooding:
  Broadcast & Multicast: enabled
  Unknown unicast: enabled
MAC aging time: 300 s, Type: inactivity
MAC limit: 4000, Action: none, Notification: syslog
MAC limit reached: no
MAC port down flush: enabled
MAC Secure: disabled, Logging: disabled
Split Horizon Group: none
Dynamic ARP Inspection: disabled, Logging: disabled
IP Source Guard: disabled, Logging: disabled
DHCPv4 snooping: disabled
IGMP Snooping: enabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: disabled
Bridge MTU: 1500
MIB cvplsConfigIndex: 1
Filter MAC addresses:
P2MP PW: disabled
Create time: 01/03/2017 11:01:11 (00:21:33 ago)
No status change since creation
ACs: 1 (1 up), VFIs: 1, PWs: 1 (1 up), PBBs: 0 (0 up)
List of ACs:
  AC: TenGigE0/2/0/1.7, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 21
    VLAN ranges: [22, 22]
    MTU 1508; XC ID 0x208000b; interworking none
    MAC learning: enabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown unicast: enabled
    MAC aging time: 300 s, Type: inactivity
    MAC limit: 4000, Action: none, Notification: syslog
    MAC limit reached: no
    MAC port down flush: enabled
    MAC Secure: disabled, Logging: disabled
    Split Horizon Group: none
    Dynamic ARP Inspection: disabled, Logging: disabled
    IP Source Guard: disabled, Logging: disabled
    DHCPv4 snooping: disabled
    IGMP Snooping: enabled
    IGMP Snooping profile: none
    MLD Snooping profile: none
    Storm Control: bridge-domain policer
    Static MAC addresses:
    Statistics:
      packets: received 714472608 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 97708776
      bytes: received 88594603392 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 12115888224
      MAC move: 0
    Storm control drop counters:
      packets: broadcast 0, multicast 0, unknown unicast 0
      bytes: broadcast 0, multicast 0, unknown unicast 0
    Dynamic ARP inspection drop counters:
      packets: 0, bytes: 0
    IP source guard drop counters:
```

```
                    packets: 0, bytes: 0
            List of VFIs:
              VFI 222 (up)
                PW: neighbor 1.1.1.1, PW ID 222, state is up ( established )
                  PW class not set, XC ID 0xc000000a
                  Encapsulation MPLS, protocol LDP
                  Source address 21.21.21.21
                  PW type Ethernet, control word disabled, interworking none
                  Sequencing not set

                  PW Status TLV in use
                    MPLS          Local                          Remote
                    ------------  ------------------------------ ------------------------
                    Label         24017                          24010
                    Group ID      0x0                            0x0
                    Interface     222                            222
                    MTU           1500                           1500
                    Control word  disabled                       disabled
                    PW type       Ethernet                       Ethernet
                    VCCV CV type  0x2                            0x2
                                  (LSP ping verification)        (LSP ping verification)
                    VCCV CC type  0x6                            0x6
                                  (router alert label)           (router alert label)
                                  (TTL expiry)                   (TTL expiry)
                    ------------  ------------------------------ ------------------------
                  Incoming Status (PW Status TLV):
                    Status code: 0x0 (Up) in Notification message
                  MIB cpwVcIndex: 3221225482
                  Create time: 01/03/2017 11:01:11 (00:21:33 ago)
                  Last time status changed: 01/03/2017 11:21:01 (00:01:43 ago)
                  Last time PW went down: 01/03/2017 11:15:21 (00:07:23 ago)
                  MAC withdraw messages: sent 0, received 0
                  Forward-class: 0
                  Static MAC addresses:
                  Statistics:
                    packets: received 95320440 (unicast 0), sent 425092569
                    bytes: received 11819734560 (unicast 0), sent 52711478556
                    MAC move: 0
                  Storm control drop counters:
                    packets: broadcast 0, multicast 0, unknown unicast 0
                    bytes: broadcast 0, multicast 0, unknown unicast 0
              DHCPv4 snooping: disabled
              IGMP Snooping profile: none
              MLD Snooping profile: none
              VFI Statistics:
                drops: illegal VLAN 0, illegal length 0
```

# Bridging on IOS XR Trunk Interfaces: Example

This example shows how to configure a as a simple L2 switch.

**Important notes:**

Create a bridge domain that has four attachment circuits (AC). Each AC is an IOS XR trunk interface (i.e. not a subinterface/EFP).

- This example assumes that the running config is empty, and that all the components are created.

- This example provides all the necessary steps to configure the to perform switching between the interfaces. However, the commands to prepare the interfaces such as no shut, negotiation auto, etc., have been excluded.

• The bridge domain is in a no shut state, immediately after being created.

• Only trunk (i.e. main) interfaces are used in this example.

• The trunk interfaces are capable of handling tagged (i.e. IEEE 802.1Q) or untagged (i.e. no VLAN header) frames.

• The bridge domain learns, floods, and forwards based on MAC address. This functionality works for frames regardless of tag configuration.

• The bridge domain entity spans the entire system. It is not necessary to place all the bridge domain ACs on a single LC. This applies to any bridge domain configuration.

• The show bundle and the show l2vpn bridge-domain commands are used to verify that the router was configured as expected, and that the commands show the status of the new configurations.

• The ACs in this example use interfaces that are in the admin down state.

Configuration Example

```
RP/0/RSP0/CPU0:router#config
RP/0/RSP0/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/5
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/6
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain test-switch
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP0/CPU0:Jul 26 10:48:21.320 EDT: config[65751]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'lab'. Use 'show configuration commit changes 1000000973'
to view the changes.
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP0/CPU0:Jul 26 10:48:21.342 EDT: config[65751]: %MGBL-SYS-5-CONFIG_I : Configured
from console by lab
RP/0/RSP0/CPU0:router#show bundle Bundle-ether10

Bundle-Ether10
  Status:                                    Down
  Local links <active/standby/configured>:   0 / 0 / 2
  Local bandwidth <effective/available>:     0 (0) kbps
  MAC address (source):                      0024.f71e.22eb (Chassis pool)
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  LACP:                                      Operational
    Flap suppression timer:                  Off
  mLACP:                                     Not configured
```

```
        IPv4 BFD:                              Not configured

        Port              Device          State        Port ID         B/W, kbps
        ----------------- --------------- ----------- -------------- ----------
        Gi0/2/0/5         Local           Configured   0x8000, 0x0001   1000000
          Link is down
        Gi0/2/0/6         Local           Configured   0x8000, 0x0002   1000000
          Link is down
RP/0/RSP0/CPU0:router#
RP/0/RSP0/CPU0:router#show l2vpn bridge-domain group examples
Bridge group: examples, bridge-domain: test-switch, id: 2000, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 4 (1 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
  List of ACs:
    BE10, state: down, Static MAC addresses: 0
    Gi0/2/0/0, state: up, Static MAC addresses: 0
    Gi0/2/0/1, state: down, Static MAC addresses: 0
    Te0/5/0/1, state: down, Static MAC addresses: 0
  List of VFIs:
RP/0/RSP0/CPU0:router#
```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

### Procedure

**Step 1**   **configure**

Enters global configuration mode.

**Step 2**   **interface Bundle-ether10**

Creates a new bundle trunk interface.

**Step 3**   **l2transport**

Changes Bundle-ether10 from an L3 interface to an L2 interface.

**Step 4**   **interface GigabitEthernet0/2/0/5**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/5.

**Step 5**   **bundle id 10 mode active**

Establishes GigabitEthernet0/2/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 6**   **interface GigabitEthernet0/2/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/6.

**Step 7**   **bundle id 10 mode active**

Establishes GigabitEthernet0/2/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 8**   **interface GigabitEthernet0/2/0/0**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/0.

**Step 9**      **l2transport**

Change GigabitEthernet0/2/0/0 from an L3 interface to an L2 interface.

**Step 10**     **interface GigabitEthernet0/2/0/1**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/1.

**Step 11**     **l2transport**

Change GigabitEthernet0/2/0/1 from an L3 interface to an L2 interface.

**Step 12**     **interface TenGigE0/1/0/2**

Enters interface configuration mode. Changes configuration mode to act on TenGigE0/1/0/2.

**Step 13**     **l2transport**

Changes TenGigE0/1/0/2 from an L3 interface to an L2 interface.

**Step 14**     **l2vpn**

Enters L2VPN configuration mode.

**Step 15**     **bridge group examples**

Creates the bridge group **examples**.

**Step 16**     **bridge-domain test-switch**

Creates the bridge domain **test-switch**, that is a member of bridge group **examples**.

**Step 17**     **interface Bundle-ether10**

Establishes Bundle-ether10 as an AC of bridge domain test-switch.

**Step 18**     **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 19**     **interface GigabitEthernet0/2/0/0**

Establishes GigabitEthernet0/2/0/0 as an AC of bridge domain **test-switch**.

**Step 20**     **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 21**     **interface GigabitEthernet0/2/0/1**

Establishes GigabitEthernet0/2/0/1 as an AC of bridge domain **test-switch**.

**Step 22**     **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 23**     **interface TenGigE0/1/0/2**

Establishes interface TenGigE0/1/0/2 as an AC of bridge domain **test-switch**.

**Step 24**     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.

# Bridging on Ethernet Flow Points: Example

This example shows how to configure a to perform Layer 2 switching on traffic that passes through Ethernet Flow Points (EFPs). EFP traffic typically has one or more VLAN headers. Although both IOS XR trunks and IOS XR EFPs can be combined as attachment circuits in bridge domains, this example uses EFPs exclusively.

**Important notes:**

- An EFP is a Layer 2 subinterface. It is always created under a trunk interface. The trunk interface must exist before the EFP is created.

- In an empty configuration, the bundle interface trunk does not exist, but the physical trunk interfaces are automatically configured. Therefore, only the bundle trunk is created.

- In this example the subinterface number and the VLAN IDs are identical, but this is out of convenience, and is not a necessity. They do not need to be the same values.

- The bridge domain test-efp has three attachment circuits (ACs). All the ACs are EFPs.

- Only frames with a VLAN ID of 999 enter the EFPs. This ensures that all the traffic in this bridge domain has the same VLAN encapsulation.

- The ACs in this example use interfaces that are in the admin down state (**unresolved** state). Bridge domains that use nonexistent interfaces as ACs are legal, and the commit for such configurations does not fail. In this case, the status of the bridge domain shows **unresolved** until you configure the missing interface.

Configuration Example

```
RP/0/RSP1/CPU0:router#configure
RP/0/RSP1/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP1/CPU0:router(config-if)#interface Bundle-ether10.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface GigabitEthernet0/6/0/5
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/6
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/7.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface TenGigE0/1/0/2.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#l2vpn
RP/0/RSP1/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP1/CPU0:router(config-l2vpn-bg)#bridge-domain test-efp
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/6/0/7.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
```

```
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP1/CPU0:router#
RP/0/RSP1/CPU0:router#show l2vpn bridge group examples
Fri Jul 23 21:56:34.473 UTC Bridge group: examples, bridge-domain: test-efp, id: 0, state:
 up, ShgId: 0, MSTi: 0
Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 3 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
  List of ACs:
    BE10.999, state: down, Static MAC addresses: 0
    Gi0/6/0/7.999, state: unresolved, Static MAC addresses: 0
    Te0/1/0/2.999, state: down, Static MAC addresses: 0
  List of VFIs:
RP/0/RSP1/CPU0:router#
```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

### Procedure

---

**Step 1**     **configure**

Enters global configuration mode.

**Step 2**     **interface Bundle-ether10**

Creates a new bundle trunk interface.

**Step 3**     **interface Bundle-ether10.999 l2transport**

Creates an EFP under the new bundle trunk.

**Step 4**     **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

**Step 5**     **interface GigabitEthernet0/6/0/5**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/5.

**Step 6**     **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 7**     **interface GigabitEthernet0/6/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/6.

**Step 8**     **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

**Step 9**     **interface GigabitEthernet0/6/0/7.999 l2transport**

Creates an EFP under GigabitEthernet0/6/0/7.

**Step 10**     **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

**Step 11**   **interface TenGigE0/1/0/2.999 l2transport**

Creates an EFP under TenGigE0/1/0/2.

**Step 12**   **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

**Step 13**   **l2vpn**

Enters L2VPN configuration mode.

**Step 14**   **bridge group examples**

Creates the bridge group named **examples**.

**Step 15**   **bridge-domain test-efp**

Creates the bridge domain named **test-efp**, that is a member of bridge group **examples**.

**Step 16**   **interface Bundle-ether10.999**

Establishes Bundle-ether10.999 as an AC of the bridge domain named **test-efp**.

**Step 17**   **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 18**   **interface GigabitEthernet0/6/0/7.999**

Establishes GigabitEthernet0/6/0/7.999 as an AC of the bridge domain named **test-efp**.

**Step 19**   **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

**Step 20**   **interface TenGigE0/1/0/2.999**

Establishes interface TenGigE0/1/0/2.999 as an AC of bridge domain named **test-efp**.

**Step 21**   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.