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.

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Virtual Private LAN Services (VPLS)	Release 7.4.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operates in native and compatibility modes. Virtual Private LAN Services (VPLS) is a class of VPN that supports the connection of multiple sites in a single bridged domain over a MPLS network.

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.

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.

Note Traffic to directly connected neighbor on EVPN or VPLS bridge won't work in the following scenarios:

- If neighbor doesn't advertise MPLS explicit null.
- If imposition node has a mix of implicit-null and labeled paths in ECMP or LFA deployment.

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.

When the number of bridge domains exceeds 200, to enable clean up and reprogramming, it takes about 120 seconds for unconfiguring L2VPN and rollback.

The following table details the minimum interval required between unconfiguring L2VPN and rollback:

Number of BDs	Minimum interval in seconds
250	180
500	300
750 or greater	600

Bridge Domain and BVI Scale

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
L2 and BVI Infrastructure	Release 7.3.1	This feature is now supported on Cisco NCS 5700 series fixed port routers and the Cisco NCS 5500 series routers that have the Cisco NC57 line cards installed and operating in the native and compatible modes.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Bridge Domain and MAC Address Scale	Release 7.4.1	This feature is now supported on Cisco NCS 5700 series fixed port routers and the Cisco NCS 5500 series routers that have the Cisco NC57 line cards installed and operating in the native mode. This feature is supported with the following scale values: • 8K Bridge Domain • 4K Pseudowires • 300K MAC addresses

The number of bridge domains (BDs) depends on the number of attachment circuits (ACs) configured per BD and also if Bridge-Group Virtual Interface (BVI) is configured or not. The number of logical interfaces (LIF) supported is less than 4000.

The following table provides an example of how the number of logical interfaces (LIF) required is calculated when two ACs are configured per BD.

Bridge Domain	Number of Bridges	AC	Total LIF required
BD with BVI	625	2	3750
BD without BVI	125	2	250
Total BD	750	-	-

Here is how the number of LIF required is calculated:

a*3+b, where a is the number of ACs with BVI and b is the number of ACs without BVI, must not exceed 4000.

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.

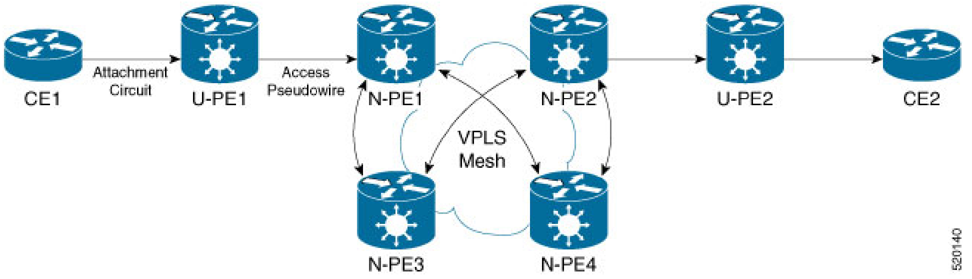
Access Pseudowire

The Access Pseudowire feature allows you to reduce the number of pseudowires (PWs) between the network Provider Edge (N-PE) devices. The user Provider Edge (U-PE) device connects to the N- PE device using access pseudowire (PW). This feature prevents signalling overhead and packet replication.

Unlike traditional VPLS where PWs terminate on a physical or logical port, an access PW terminates on N-PE devices. For each VPLS service, create an access PW between U-PEs and N-PEs.

VPLS requires a full mesh of pseudowire (PWs) between L2VPN PEs that participate in the VPLS service. For each VPLS service, PWs must be set up between the PEs. In a full mesh of PWs, the number of PWs increases as the number of PEs increases causing scalability issues. You can decrease the number of PWs with a hierarchy of PEs.

Figure 1: Access Pseudowire



In this topology, a user Provider Edge (U-PE) device has ACs to the CEs. The U-PE device transports the CE traffic over an access PW to a network Provider Edge (N-PE) device. The N-PE is a core VPLS PE connected with other N-PEs in a VPLS mesh. On the N-PE, the access PW coming from the U-PE is much like an AC. The U-PE is not part of the mesh with the other N-PEs. So the N-PE considers the access PW as an AC. The N-PE forwards traffic from that access PW to the core PWs that are part of the VPLS full mesh. Configure the core PWs between N-PEs under a VFI. Apply the split horizon rule to all the core PWs configured under the VFI. Access PWs from U-PEs are not configured under a VFI, so they do not belong to the same Split Horizon Groups (SHGs) as the VFI PWs. Traffic is forwarded from an access PW to a VFI PW and conversely.

You must configure the access pseudowire in a split-horizon group.

Configure Access Pseudowire

Perform this task to configure Access Pseudowire feature.

```
/* Configure U-PE1 */
Router#configure
Router(config)# interface TenGigE0/1/0/5.2 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 2
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/5.2
Router(config-l2vpn-xc-p2p)# neighbor 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

```
/* Configure N-PE1 */
Router#configure
Router(config)l2vpn
Router(config-l2vpn)#router-id 172.16.0.1
Router(config-l2vpn)#pw-class class1
Router(config-l2vpn-pwc)#encapsulation mpls
Router(config-l2vpn-pwc-mpls)#transport-mode ethernet
Router(config-l2vpn-pwc-mpls)#exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet0/1/0/3.2
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi vfi1
Router(config-l2vpn-bg-bd-vfi)#neighbor 10.0.0.1 pw-id 2
Router(config-l2vpn-bg-bd-vfi-pw)#pw-class class1
Router(config-l2vpn-bg-bd-vfi-pw-pw)#commit
```

Running Configuration

This sections shows Access Pseudowire running configuration.

```
/* On U-PE1 */
configure
 interface TenGigE0/1/0/5.2
  encapsulation dot1q 2
  rewrite ingress tag pop 1 symmetric
!
l2vpn
 xconnect group XCON1
  p2p xc1
   interface TenGigE0/1/0/5.2
   neighbor 172.16.0.1 pw-id 1
   !
!
--------------------------------------------------------------------------
/* On N-PE1 */
```

```
l2vpn
 router-id 172.16.0.1
 pw-class class1
 encapsulation mpls
 transport-mode ethernet
 !
!
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface GigabitEthernet0/1/0/3.2
    split-horizon group
   !
 !
!
vfi vf1
 neighbor 10.0.0.1 pw-id 2
  pw-class class1
  !
!
```

Verification

Verify Access Pseudowire configuration.

```
Router:U-PE1#show l2vpn xconnect group XCON1
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready, (PP) = Partially Programmed

XConnect                   Segment 1                       Segment 2
Group      Name       ST   Description            ST       Description      ST
-----------------   ----- --------------   -----------------------------   ------
XCON_1     xc1        UP   Te0/1/0/5.2            UP       172.16.0.1   1   UP
-----------------------------------------------------------------------------------
```

```
Router:N-PE1#show l2vpn bridge-domain bd1
PW: neighbor 10.0.0.1, PW ID 2, state is up ( established )
      PW class mpls, XC ID 0xc0000008
      Encapsulation MPLS, protocol LDP
      Source address 172.16.0.1
      PW type Ethernet, control word disabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
      LSP : Up

      PW Status TLV in use
        MPLS         Local                          Remote
        ------------ ------------------------------ ---------------------------
        Label        24752                          24752
        Group ID     0x2                            0x2
        Interface    Access PW                      Access PW
        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)
        ------------ ------------------------------ ---------------------------
```

Related Topics

- Access Pseudowire, on page 4

Associated Commands

- show l2vpn xconnect group
- show l2vpn bridge-domain

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
- Storm control counters are not supported.

The storm control that is applied to multiple subinterfaces of the same physical port pertains to that physical port only. All subinterfaces with storm control configured are policed as aggregate under a single policer rate shared by all EFPs. None of the subinterfaces are configured with a dedicated policer rate. When a storm occurs on several subinterfaces simultaneously, and because subinterfaces share the policer, you can slightly increase the policer rate to accommodate additional policing.

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Ingress Storm control (BUM) policers	Release 7.4.2	This feature monitors ingress traffic to prevent outages caused by a broadcast, unregistered-multicast, or unknown-unicast (BUM) traffic storm. This feature is now supported on routers that have Cisco NC57 line cards with external TCAM that are installed and operating in native mode.

Note From Cisco Release 7.4.2, in NC57 Line cards with external TCAM operate in native mode. L2 storm control is independently supported on L2 main interface or L2 sub interfaces that can be attached as a part of bridge domain.

These features are supported:

- Bridging IOS XR Trunk Interfaces
- Bridging on EFPs

Storm Control on Bridge Domain

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
Storm Control Configuration for Subinterfaces	Release 7.8.1	Storm control helps prevent LAN ports from being disrupted by a broadcast, multicast, or unicast traffic storm. You can now configure different storm control rates for each subinterface on a physical port. This will give you control at a granular level and prevent flooding of excess traffic at the subinterface level. In earlier releases, storm control could be configured only at the physical port level or only on one subinterface under a main interface. This feature modifies the **hw-module storm-control-combine-policer-bw enable** command to enable per subinterface configuration support for storm control.

Storm Control provides Layer 2 port security under a Virtual Private LAN Services (VPLS) bridge by preventing excess traffic from disrupting the bridge.

A traffic storm occurs when packets flood a VPLS bridge, creating excessive traffic and degrading network performance. Storm control prevents VPLS bridge disruption by suppressing traffic when the number of packets reaches configured threshold levels. You can configure separate threshold levels for different types of traffic on an access circuit (AC) under a VPLS bridge.

Storm control monitors incoming traffic levels on a port or a subinterface, and drops traffic when the number of packets reaches the configured threshold level during any 1-second interval. The 1-second interval is set in the hardware and is not configurable. The number of packets allowed to pass during this interval is configurable, per subinterface, per port, per traffic type. During this interval, the traffic level is compared with the configured storm control level. When the incoming traffic reaches the storm control level configured on the bridge port, storm control drops traffic until the end of storm control interval. At the beginning of a new interval, traffic of the specified type is allowed to pass on the port. The thresholds are configured using a packets per second (pps) and kilobit per second (kbps) rate.

Storm control has little impact on router performance. Packets passing through ports are counted regardless of whether the feature is enabled. Additional counting occurs only for the drop counters, which monitor dropped packets. Storm control counts the number of packets dropped per port. The drop counters are cumulative for all traffic types.

Supported Traffic Types for Storm Control

On each VPLS bridge port, you can configure up to three storm control thresholds—one for each of the supported traffic types. If you do not configure a threshold for a traffic type, then storm control is not enabled on that port or interface for that traffic type.

The supported traffic types are:

- Broadcast traffic—Packets with a packet destination MAC address equal to FFFF.FFFF.FFFF.
- Multicast traffic—Packets with a packet destination MAC address not equal to the broadcast address, but with the multicast bit set to 1. The multicast bit is bit 0 of the most significant byte of the MAC address.
- Unknown unicast traffic—Packets with a packet destination MAC address not yet learned.

Restrictions for Storm Control

- On multi NPU systems, the storm-control policer is applied per NPU core.
- Only BUM traffic ingress from ACs will be subjected to storm-control policers. BUM traffic ingressing on VPLS PW and EVPN core will not be subjected to storm-control policers.
- Storm-control configuration is not supported on both Bridge-domain and Bridge-port.

Configure Storm Control on Bridge Domain

You can configure storm control on a physical port or on a subinterface. The storm control rates that are configured on a subinterface is applied to all the subinterfaces in the main port.

The thresholds are configured using packets per second (pps) or kilobit per second (kbps) rate.

Configuration Example

1. Create a bridge group with bridge domain.
2. Assign an interface or subinterface to the bridge domain.
3. Configure storm control for the interface or subinterface.

The following example shows storm control configured for broadcast traffic type on an interface:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg0
Router(config-l2vpn-bg)# bridge-domain bd0
Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/1
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 4500
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-ac)# exit
```

Running Configuration

```
configure
 l2vpn
  bridge group bg0
```

```
   bridge-domain bd0
    interface HundredGigE0/0/0/1
     storm-control broadcast pps 4500
    !
```

Verification

The following example shows a truncated output.

```
Router# show l2vpn bridge-domain bd-name bd0 detail
Legend: pp = Partially Programmed.
Bridge group: bg0, bridge-domain: bd0, id: 0, state: up, ShgId: 0, MSTi: 0

  ...................

  No status change since creation
  ACs: 1 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: HundredGigE0/0/0/1, state is unresolved
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 131072, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control:
        Broadcast: enabled(4500 pps)
        Multicast: disabled
        Unknown unicast: disabled
      Static MAC addresses:
      PD System Data: Learn key: 0
```

Configure Storm Control per Subinterface

Starting from Release 7.8.1, you can enable per subinterface configuration support for storm control by using the **hw-module storm-control-combine-policer-bw enable** command. For more information, see Combined Policer Mode, on page 14.

By default, when storm control is configured on a subinterface, the same configuration is applied to all the subinterfaces in that main port.

Configuration Example

1. Configure combined policer to enable the storm control configuration on subinterfaces.
2. Create a bridge group with bridge domain.
3. Assign a subinterface to the bridge domain.

4. Configure storm control for the subinterface.
5. Assign another subinterface to the bridge domain and configure storm control with different parameters for the subinterface.

You can repeat the configuration of storm control on different subinterfaces of the same main port.

```
/* Configure combined policer */

Router# configure
Router(config)# hw-module storm-control-combine-policer-bw enable
```

Note You must manually reload the router to activate the **hw-module storm-control-combine-policer-bw enable** command.

```
/* Create a bridge group */

Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
```

```
/* Assign a subinterface and configure storm control */

Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/1.10
Router(config-l2vpn-bg-bd-ac)# storm-control unknown-unicast pps 500
Router(config-l2vpn-bg-bd-ac)# storm-control multicast pps 2000
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 1000
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-ac)# exit
```

```
/* Assign another subinterface and configure storm control with different parameters */

Router(config-l2vpn-bg-bd)# interface HundredGigE0/0/0/1.20
Router(config-l2vpn-bg-bd-ac)# storm-control unknown-unicast pps 200
Router(config-l2vpn-bg-bd-ac)# storm-control multicast pps 1000
Router(config-l2vpn-bg-bd-ac)# storm-control broadcast pps 2000
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-ac)# exit
```

Running Configuration

```
configure
 hw-module storm-control-combine-policer-bw enable
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface HundredGigE0/0/0/1.10
     storm-control unknown-unicast pps 500
     storm-control multicast pps 2000
     storm-control broadcast pps 1000
    !
    interface HundredGigE0/0/0/1.20
     storm-control unknown-unicast pps 200
     storm-control multicast pps 1000
     storm-control broadcast pps 2000
    !
```

Verification

The following example shows a truncated output.

```
Router# show l2vpn bridge-domain bd-name bd1 detail
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd1, id: 1, state: up, ShgId: 0, MSTi: 0

 ..............

  ACs: 2 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: HundredGigE0/0/0/1.10, state is unresolved
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 131072, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control:
        Broadcast: enabled(1000 pps)
        Multicast: enabled(2000 pps)
        Unknown unicast: enabled(500 pps)
      Static MAC addresses:
      PD System Data: Learn key: 0

    AC: HundredGigE0/0/0/1.20, state is unresolved
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 131072, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control:
        Broadcast: enabled(2000 pps)
        Multicast: enabled(1000 pps)
        Unknown unicast: enabled(200 pps)
      Static MAC addresses:
      PD System Data: Learn key: 0
```

Combined Policer Mode

Combined policer mode is introduced in NC57 line cards operating in native mode to conserve the policer resources used which helps in increasing the storm control policer scale.

In this mode when more than one policer is defined under a single attachment circuit [AC], then the result of policer is sum of all the policers applied under that AC. Layer 2 storm control feature is applicable only for the BUM traffic type in that particular AC.

Use the **hw-module storm-control-combine-policer-bw enable** command to enable the policer mode.

Note You must manually reload the router to activate the **hw-module storm-control-combine-policer-bw** command.

With the combined mode policer profile, any policer type configured is applicable for any of the *broadcast*, *unknown-unicast*, or *multicast* traffic under that AC and all are rate limited up to the policer configured value.

For example, *broadcast* policer is configured at 1000 kbps will rate limit any of the *broadcast*, *unknown-unicast*, *multicast* to a maximum of 1000 kbps combined. *Broadcast* policer configured at 1000 kbps in addition to the multicast policer configured at 2000 kbps, will rate limit any of the *broadcast*, *unknown-unicast*, *multicast* to the maximum of 3000 kbps combined.

Starting from Release 7.8.1, you can use the **hw-module storm-control-combine-policer-bw enable** command to enable storm control configuration per subinterface. When you configure storm control on the subinterfaces, for each subinterface configuration, a policer ID is derived from the logical interface. The multicast destination will derive the policer ID for further processing of packets.

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
```

VPLS VFI with BVI as Routed Interface

Table 6: Feature History Table

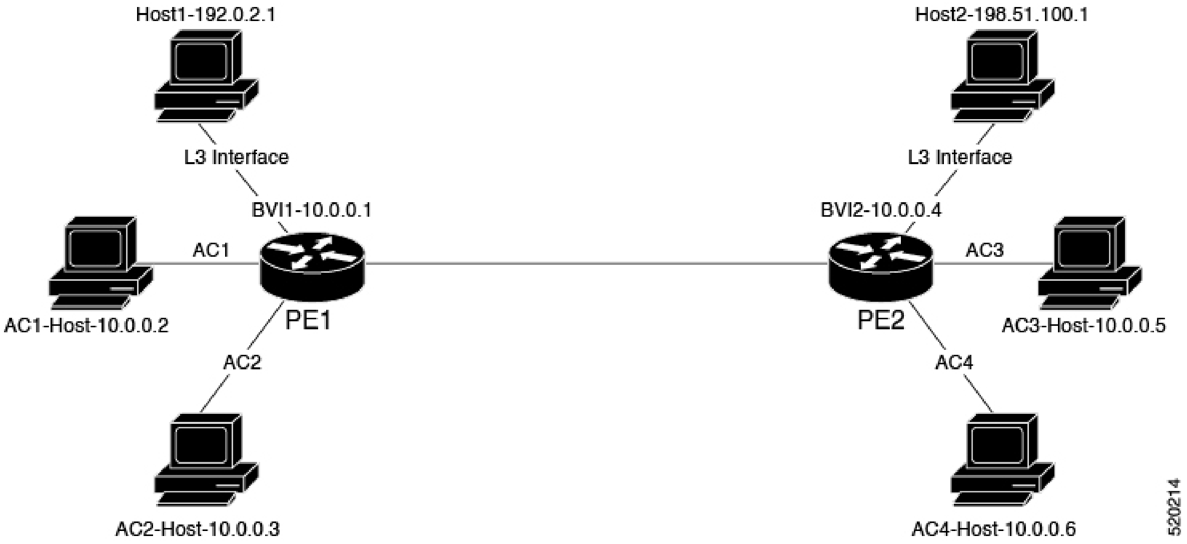
Feature Name	Release Information	Feature Description
VPLS VFI with BVI as Routed Interface on Cisco 5700 Fixed Routers and Cisco NC57 Line Cards	Release 7.7.1	VPLS virtual forwarding interface (VFI) is a multipoint Layer 2 VPN technology connecting two or more customer devices to perform native bridging functions such as forwarding. Bridged Virtual Interface (BVI) is a virtual interface within the router that acts like a normal routed interface. You can enable BVI on VPLS VFI, where multiple interfaces can be part of a single bridge group. This functionality allows you to route the incoming traffic to the bridge group to L3 interfaces. This feature is now supported on Cisco 5700 Fixed Routers and on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

The VPLS VFI with BVI as Routed Interface feature allows you to route the VPLS PW traffic over the BVI interface.

Integrated routing and bridging (IRB) enables you to route the packets in and out of a bridge domain using a Bridge-Group Virtual Interface (BVI). The BVI is a virtual interface configured on the router. It acts as a gateway routed interface towards the core network.

Configure a BVI on a single bridge domain that represents the link between the bridging and the routing domains on the router. To receive the packets from a bridged interface that are destined to a routed interface, configure BVI with the appropriate IP address, which is in the same subnet as the hosts in the bridge domain.

Figure 2: VPLS VFI with BVI as Routed Interface



This topology explains two types of traffic flow:

- Routed local traffic: Consider a traffic flow from AC1 Host to Host1. AC1 Host sends the traffic to BVI1. Attach AC1 Host and BVI1 to the same bridge domain of PE1. PE1 routs the traffic through BVI1 and sends it to Host1. L3 interface connects Host1 and PE1.
- Routed remote traffic: Consider a traffic flow from AC2 Host to Host2. AC2 Host sends the traffic to the bridge domain of PE1. PE1 sends the traffic to BVI2. AC2 Host is part of BVI2 subnet. PW sends the traffic to the bridge domain of PE2. PE2 routes the traffic through BVI2 and sends it to Host2. L3 interface connects Host2 and PE2.

Restrictions

- The following protocols are not supported when the bridge domain is attached with both PW and BVI: DHCP, ERPS, CDP, HSRP, IGMP Snooping, VRRP, CFM, LACP, and BFDoBVI.

Configure VPLS VFI with BVI as Routed Interface

Perform this task to route the VPLS PW traffic dynamically over the BVI interface.

Configuration Example

```
/* PE1 Configuration */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/0.1 -> AC1-L2 Sub-Interface (AC)
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi core
Router(config-l2vpn-bg-bd-vfi)#neighbor 209.165.200.225 pw-id 1 -> VPLS Core-PW
Router(config-l2vpn-bg-bd-vfi-pw)#exit
Router(config-l2vpn-bg-bd-vfi)#exit
Router(config-l2vpn-bg-bd)#routed interface BVI1 -> BVI-1 Interface
Router(config-l2vpn-bg-bd-bvi)#root
Router(config)#interface BVI1
Router(config-if)#ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)#commit


/* PE2 Configuration */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/1.1 -> AC3 L2 subinterface(AC)
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi core
Router(config-l2vpn-bg-bd-vfi)#neighbor 209.165.200.226 pw-id 1 -> VPLS Core-PW
Router(config-l2vpn-bg-bd-vfi-pw)#exit
Router(config-l2vpn-bg-bd-vfi)#exit
Router(config-l2vpn-bg-bd)#routed interface BVI2 -> BVI-2 Interface
Router(config-l2vpn-bg-bd-bvi)#root
Router(config)#interface BVI2
Router(config-if)#ipv4 address 10.0.0.4 255.0.0.0
Router(config-if)#commit
```

Running Configuration

This section shows VPLS VFI with BVI as Routed Interface configuration.

```
/* PE1 Configuration */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface TenGigE0/0/0/0.1 -> AC1-L2 Sub-Interface (AC)
    !
    vfi core
     neighbor 209.165.200.225 pw-id 1 -> VPLS Core-PW
     !
    !
    routed interface BVI1 -> BVI-1 Interface
    !
  !
interface BVI1
 ipv4 address 10.0.0.1 255.0.0.0


/* PE2 Configuration */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd2
    interface TenGigE0/0/0/1.1 -> AC3 L2 Sub-Interface (AC)
    !
    vfi core
     neighbor 209.165.200.226 pw-id 1 -> VPLS Core-PW
     !
    !
    routed interface BVI2 -> BVI2 Interface
    !
  !
interface BVI2
 ipv4 address 10.0.0.4 255.0.0.0
```

Verification

Verify that you have configured the VPLS VFI with BVI as Routed Interface feature successfully.

```
Router-PE1#show l2vpn bridge-domain neighbor 209.165.200.225 detail
Legend: pp = Partially Programmed.
Bridge group: 1, bridge-domain: 1, id: 0, state: up, ShgId: 0, MSTi: 0
  VINE state: BVI Resolved
  MAC learning: enabled
  MAC withdraw: enabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  Create time: 10/01/2020 04:18:29 (00:14:06 ago)
  ACs: 2 (2 up), VFIs: 1, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of Access PWs:
  List of VFIs:
    VFI 1 (up)
      PW: neighbor 209.165.200.225, PW ID 1, state is up ( established )
        PW class mpls, XC ID 0xc0000002
        Encapsulation MPLS, protocol LDP
        Source address 209.165.200.226
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set
        LSP : Up

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ -------------------------
          Label        24006                          24002
          Group ID     0x0                            0x0
          Interface    1                              1
          MTU          1500                           1500
          Control word disabled                       disabled

          ------------ ------------------------------ -------------------------
```

Related Topics

- VPLS VFI with BVI as Routed Interface, on page 15

Associated Commands

- show l2vpn bridge-domain detail

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.

The range of MAC address aging time is from 300 seconds to 30,000 seconds. The maximum MAC address aging time among all bridges is considered for calculating the age. You cannot configure the MAC address aging time on each AC or PW interface. Configure MAC address aging time in the bridge domain configuration mode. There is no show command to display the highest MAC address aging time.

MAC Address Limit

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Configure MAC Address Limit for Bridge Domains to Learn Static Addresses	Release 7.8.1	You can now configure the MAC address limit for bridge domains to learn only static MAC addresses and to drop traffic from unknown sources. Malicious attackers can spoof a Layer 2 MAC address to change dynamic entries in the MAC table. However, with this functionality enabling you to configure the MAC address limit for bridge domains to learn only static MAC addresses, the dynamic MAC addresses are blocked. In addition, a static entry always overrules dynamic entries. This functionality thus prevents the interception of your data by unauthorized users and improves your network security.

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 you configure MAC address limit for a bridge-domain, the following MAC addresses are considered by the network:

- Dynamic MAC addresses that are learned automatically via an ARP request.
- Static MAC addresses that are manually configured.
- EVPN Sync MAC addresses that are MAC entries synchronized across EVPN PEs.
- EVPN Remote MAC addresses that are assigned to remote EVPN PEs.

Restrictions

- You can configure up to a maximum of six different mac-limit values under a bridge domain for the following routers and line cards:
  - NCS-55A1-24H
  - NCS-55A1-48Q6H
  - NCS-55A1-36H
  - NCS-55A1-36H-SE
  - NCS-55A2-MOD-HD-S

  - NCS-55A2-MOD-S
  - NCS-5502
  - NCS-5502-SE
  - NCS55-36x100G-S
  - NC55-24H12F-SE
  - NCS55-36x100G-A-SS

- You can configure up to a maximum of 30 different mac-limit values under a bridge domain on routers that have the Cisco NC57 line cards installed.
- For NCS55xx routers and NCS57 line cards, the mac-limit value programmed in the hardware depends on the:
  - Static MAC address configured under the AC for a bridge domain.
  - BVI configured under a bridge domain.

Depending on the BVI or static MAC address configured, new mac-limit profiles are required. The following example shows the different bridge domains with default mac-limit with static MAC address and BVI.

Example 1

In this example, the bridge domain requires a default mac-limit profile. For instance, default mac-limit = X.

```
bridge-domain 1
 interface HundredGigE 0/0/0/10
```

Example 2

In this example, the bridge domain requires a new mac-limit profile with mac-limit = X+1 to accommodate the static BVI MAC address.

```
bridge-domain 2
 interface HundredGigE 0/0/0/11
  routed interface bvi
```

Example 3

In this example, the bridge domain requires a new mac-limit profile with mac-limit = X+2 to accommodate two static MAC addresses configured under the AC.

```
bridge-domain 3
 interface HundredGigE 0/0/0/12
  static-mac-address 0000.1111.2222
  static-mac-address 0000.2222.1111
```

Configure MAC Address Limit

Configure the MAC address limit using the **maximum** command. The MAC address learning is restricted to the configured limit.

When the number of learned MAC addresses reaches the configured limit, you can configure the bridge behavior by using the **action** command. You can configure the action to perform one of the following:

- **flood**: All the unknown unicast packets, with unknown destinations addresses, are flooded over the bridge.

- **no-flood**: All the unknown unicast packets, with unknown destination addresses, are dropped.
- **shutdown** : All the packets are dropped.

When the MAC limit is exceeded, use the **notification {both | none | trap}** command to send notifications in one of the following forms:

- **trap**: Sends Simple Network Management Protocol (SNMP) trap notification.
- **both**: Sends both syslog and trap notifications.
- **none**: No notifications are sent.

By default, syslog message is sent.

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.

Configuration Example

In this example, MAC address limit is configured as 5000 and MAC limit action is set to flood the packets. As notification is not configured, syslog entries are sent when the MAC limit is exceeded.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg-0
Router(config-l2vpn-bg)# bridge-domain bd-0
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 5000
Router(config-l2vpn-bg-bd-mac-limit)# action flood
```

Verification

Use the **show l2vpn bridge-domain** command to view the MAC address limit configuration.

```
Router# show l2vpn bridge-domain bd-name bd-0 detail
Legend: pp = Partially Programmed.
Bridge group: bg-0, bridge-domain: bd-0, id: 25, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN Native
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 5000, Action: flood, Notification: syslog
  MAC limit reached: no, threshold: 80%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
```

Configure MAC Address Limit for Static MAC Addresses

The dynamic MAC addresses are not learned when the MAC address limit is configured to be less than the number of static MAC address entries. When you configure the MAC address limit as zero using the **maximum** command, the dynamic MAC addresses are blocked and only static MAC addresses are learned.

To discard the traffic arriving from an unknown source, set the MAC limit action as **no-flood**, so that all the unknown unicast, broadcast, and multicast packets are dropped.

Configuration Example

In this example, MAC address limit is configured as zero and MAC limit action is set to **no-flood**.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg-0
Router(config-l2vpn-bg)# bridge-domain bd-0
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# limit
Router(config-l2vpn-bg-bd-mac-limit)# maximum 0
Router(config-l2vpn-bg-bd-mac-limit)# action no-flood
```

Verification

Use the **show l2vpn bridge-domain** command to view the MAC address limit configuration.

```
Router# show l2vpn bridge-domain bd-name bd-0 detail
Legend: pp = Partially Programmed.
Bridge group: bg-0, bridge-domain: bd-0, id: 25, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: EVPN Native
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 0, Action: no-flood, Notification: syslog
  MAC limit reached: no, threshold: 80%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
```

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)# 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.

Flooding Disable

The Flooding Disable feature prevents forwarding of Broadcast, Unknown-unicast and Multicast (BUM) traffic on the bridge domain. You can disable flooding of BUM traffic at the bridge level or at the interface level. By disabling flooding at the bridge level, you can prevent forwarding of BUM traffic on attachment circuit (AC), pseudowire (PW), and EVPN LIFs.

You can also disable only unknown unicast traffic at the bridge level or at the interface level. By disabling flooding of unknown unicast traffic at the bridge level, you can prevent forwarding of unknown unicast traffic on attachment circuit (AC), pseudowire (PW), and EVPN LIFs.

By disabling flooding of unknown unicast traffic at the interface level, you can prevent forwarding of unknown unicast traffic on AC alone.

Configure Flooding Disable

Perform this task to configure Flooding Disable feature.

You can disable flooding of:

- BUM traffic at the bridge level
- Unknown-unicast traffic at the bridge level
- Unknown-unicast traffic at the interface level

However, the flooding disable of unknown-unicast traffic at the bridge level takes effect only when the **flooding disable** command is not configured for BUM traffic at the bridge level.

The flooding disable of unknown-unicast traffic at the interface level takes effect only when **flooding disable** and **flooding unknown-unicast disable** commands are not configured at the bridge level.

Configuration Example

```
/* Configuration to disable flooding of BUM traffic at the bridge level */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#flooding disable\
Router(config-l2vpn-bg-bd)#commit

/* Configuration to disable flooding of unknown-unicast traffic at the bridge level */
Router# configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#flooding unknown-unicast disable
Router(config-l2vpn-bg-bd)#commit

/* Configuration to disable flooding of unknown-unicast traffic at the interface level */
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain bd1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/0.2
Router(config-l2vpn-bg-bd-ac)#flooding unknown-unicast disable
Router(config-l2vpn-bg-bd-ac)#commit
```

Running Configuration

This section shows flooding disable running configuration.

```
/* Configuration to disable flooding of BUM traffic at the bridge level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding disable
    flooding unknown-unicast disable
    interface TenGigE0/0/0/0.2
      flooding unknown-unicast disable
   !

/* Configuration to disable flooding of unknown-unicast traffic at the bridge level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    flooding unknown-unicast disable
    !
  !


/* Configuration to disable flooding of unknown-unicast traffic at the interface level */
configure
 l2vpn
  bridge group bg1
   bridge-domain bd1
    interface TenGigE0/0/0/0.2
      flooding unknown-unicast disable
      !
    !
  !
```

Associated Commands

- flooding disable
- flooding unknown-unicast disable

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 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.

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.

Note MAC Address Limit action is supported only on the ACs and not on the core pseudowires.

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 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. Aging time is counted from the last time that the switch saw the MAC address. The range of MAC address aging time is from 300 seconds to 30,000 seconds. 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.

MAC Address Withdrawal

The MAC Address Withdrawal feature provides faster convergence by removing MAC addresses that are dynamically learned. This feature uses Label Distribution Protocol (LDP)-based MAC address withdrawal message. A MAC list Type Length Value (TLV) is part of the MAC address withdrawal message.

This feature also supports optimization of MAC address withdrawal. The optimization allows PEs to retain the MAC addresses that are learned from the CE devices over the access side. Only MAC addresses that are learned from peer PEs are flushed out. This avoids unnecessary MAC flushing toward attachment circuit (AC) side and ensures better utilization of bandwidth and resources.

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

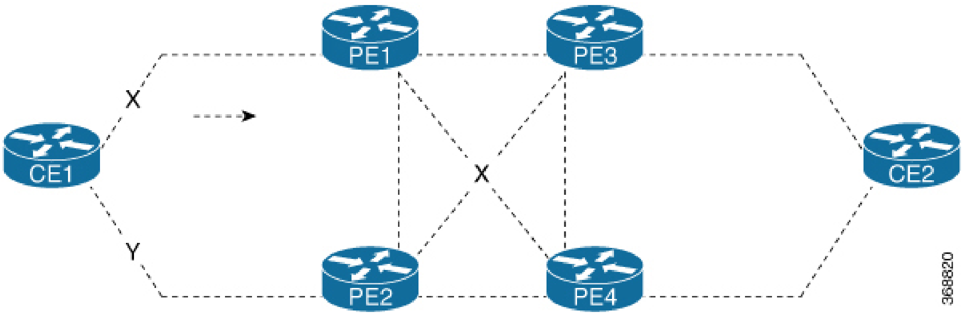
Topology

Consider the following topology in which CE1 is dual-homed to PE1 and PE2. The link X actively participates in VPLS while Y is a redundant link. Initially PE1, PE2, PE3, and PE4 learn their MAC forwarding tables that are based on the traffic profile and traffic becomes a known unicast. When the MAC address withdrawal feature is enabled on all PEs, PEs delete MAC entries when they receive MAC address withdrawal message. The following are the MAC address withdrawal messages that are based on the status of link:

- Scenario 1: When link X, which is the AC of PE1 goes down, PE1 sends an LDP MAC withdrawal TLV message “FLUSH ALL MAC FROM ME” to neighbor PEs. Peer PEs delete MAC addresses that are learned only from PE1. PE2, PE3, and PE4 flush only MAC addresses that are learned from PE1. The PE1 initiates MAC flush when its access side AC goes down.

- Scenario 2: When link Y, which is the AC of PE2 comes up, PE2 sends an LDP MAC withdrawal TLV message “FLUSH ALL MAC BUT ME” to neighbor PEs. Peer PEs flush all MAC addresses except those from the PE which receives the request.

Figure 3: MAC Address Withdrawal



Restrictions

To configure MAC address withdrawal, the following restrictions are applicable:

- This feature is not supported over H-VPLS network.
- This feature is not supported over BGP signaling and discovery.
- MAC withdraw relaying is not supported.

Configure MAC Address Withdrawal

Configuration Example

Perform this task to configure MAC address withdrawal.

```
/* Configure MAC address withdrawal on PE1. This configuration is required for scenario 1
*/
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw state-down
Router(config-l2vpn-bg-bd-mac)# exit
Router(config-l2vpn-bg-bd)# interface tenGigE0/0/0/0
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# vfi vf1
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.1 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# commit

/* Configure optimization of MAC address withdrawal on PE1. This configuration is required
 for scenario 1  */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw optimize
Router(config-l2vpn-bg-bd-mac)# exit
Router(config-l2vpn-bg-bd)# neighbor 192.0.2.1 pw-id 1234
Router(config-l2vpn-bg-bd-pw)# exit
Router(config-l2vpn-bg-bd)# vfi vf1
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.2 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# exit
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.3 pw-id 2
Router(config-l2vpn-bg-bd-vfi-pw)# commit

/* MAC address withdrawal is enabled by default when AC comes up. Use the following
configuration if you want to disable MAC address withdrawal. This configuration is required
 for scenario 2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# withdraw disable
Router(config-l2vpn-bg-bd-mac)# commit
```

Running Configuration

This section shows the running configuration of MAC address withdrawal.

```
/* Configure MAC address withdrawal on PE1 */
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw state-down
   !
   interface tengige 0/0/0/0
   !
   vfi vf1
    neighbor 192.0.2.1 pw-id 1
    !

/* Configure optimization of MAC address withdrawal on PE1 */
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw optimize
   !
   neighbor neighbor 192.0.2.1 pw-id 1234
   !
   vfi vf1
    neighbor neighbor 192.0.2.2 pw-id 1
    !
    neighbor neighbor 192.0.2.3 pw-id 2


/* Disable MAC address widhrawal on PE2 */
l2vpn
 bridge group bg1
  bridge-domain bd1
   mac
    withdraw disable
   !
```

Verification

Verify MAC address withdrawal configuration.

```
/* Verify if MAC address widhrawal is configured on PE1 */
Router:PE1# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port down

/* Verify if optimization of MAC address withdrawal is configured on PE1 */
Router:PE1# show l2vpn bridge-domain detail
MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port down (optimization)
```

Related Topics

- MAC Address Withdrawal, on page 47

Associated Commands

- mac withdraw
- show l2vpn bridge-domain detail

MAC Loop Prevention

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
MAC Loop Prevention	Release 7.5.2	This feature helps reduce network congestion and avoid traffic loss by shutting down a port after it reaches the configured number of MAC moves within the specified move interval. You can configure this feature at the bridge-domain level using the **mac secure** command. This feature is now supported on routers that have Cisco NC57 line cards installed and operate in native and compatibility modes.

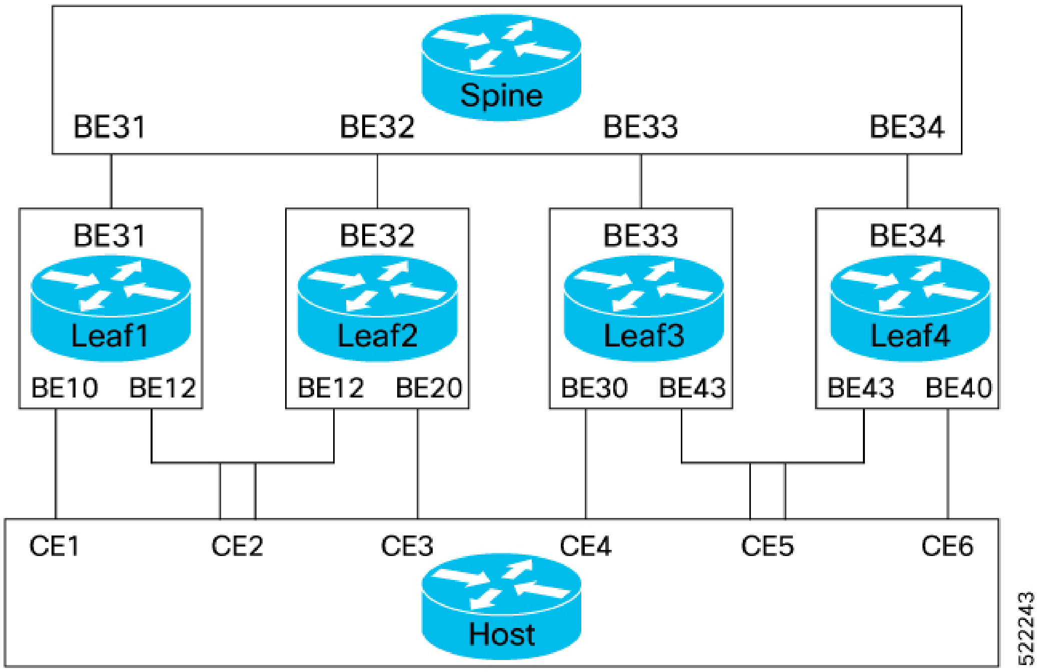
Note This feature is applicable only to the Distributed Disaggregated Chassis (DDC).

In case of network instability like an interface flap, a MAC address might be learned from a new interface. This is normal network convergence, and the mac-address-table is updated dynamically. A MAC move occurs when the same MAC address is learned on multiple interfaces. However, constant MAC moves often indicate network instability during an L2 loop. This feature lets you report MAC moves and take corrective actions such as shutting down an offending port.

The MAC Loop Prevention feature allows you to shut down the port after it exceeds the configured number of MAC moves within the specified move interval. You can configure this feature at the bridge-domain level using the **mac secure** command. The default number of MAC moves is five times for a move interval of 180 seconds. If the number of MAC moves exceeds the configured value, the MAC entry is marked as duplicate and the port is shut down. This feature helps you to reduce network congestion and avoid traffic loss. This feature is supported on physical and bundle AC, PW, and EVPN.

You can recover the shutdown port after a particular time by using the **shutdown-recovery-timeout** command after which the port automatically becomes active. If the recovery time is not configured, the shutdown port is recovered after three times of move interval. For example, if the move interval is 30 seconds, the shutdown port becomes active after 90 seconds.

Let’s see how this feature works in the following scenarios:



- MAC learning within the node - When Leaf1 learns the same MAC address on both the interfaces, BE10 and BE12, the MAC is marked as duplicate and the port is shutdown after it exceeds the configured number of MAC moves within specified interval. For example, consider the MAC move count is configured as 5 for 180 seconds. If the traffic flows starts from BE10, and, the configured MAC move count ends at interface BE12, the port at the interface BE12 is shutdown.
- MAC learning between the nodes - When the same MAC address is learnt on BE10 and 20, the MAC is marked as duplicate and the port is shutdown after it exceeds the configured number of MAC moves within specified interval. For example, consider the MAC move count is configured as 5 for 180 seconds. If the traffic starts from BE10, the configured MAC move count ends at interface BE20, the port at the interface BE20 is shutdown.

Note When a Leaf that is enabled with this feature receives two MACs from the Leafs that are not enabled with this feature, this feature won't take effect.

Configuration Example

Perfrom this task on a Leaf to configure MAC loop prevention.

```
/* MAC Loop Prevention for VPLS */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge-group BG1
Router(config-l2vpn-bg)#bridge-domain BD1
Router(config-l2vpn-bg-bd)#mac secure
Router(config-l2vpn-bg-bd-mac-sec)#action shutdown
Router(config-l2vpn-bg-bd-mac-sec)#threshold
Router(config-l2vpn-bg-bd-mac-sec)#shutdown-recovery-timeout 300
Router(config-l2vpn-bg-bd-mac-sec)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.1
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.2
Router(config-l2vpn-bg-bd-ac)#commit

/* MAC Loop Prevention for PW */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge-group BG1
Router(config-l2vpn-bg)#bridge-domain BD1
Router(config-l2vpn-bg-bd)#mac secure
Router(config-l2vpn-bg-bd-mac-sec)#action shutdown
Router(config-l2vpn-bg-bd-mac-sec)#threshold
Router(config-l2vpn-bg-bd-mac-sec)#shutdown-recovery-timeout 300
Router(config-l2vpn-bg-bd-mac-sec)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.3
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#vfi VFI1
Router(config-l2vpn-bg-bd-vfi)#neighbor 192.168.0.4 pw-id 3
Router(config-l2vpn-bg-bd-vfi-pw)#commit

/* MAC Loop Prevention for EVPN */
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge-group BG1
Router(config-l2vpn-bg)#bridge-domain BD1
Router(config-l2vpn-bg-bd)#mac secure
Router(config-l2vpn-bg-bd-mac-sec)#action shutdown
Router(config-l2vpn-bg-bd-mac-sec)#threshold
Router(config-l2vpn-bg-bd-mac-sec)#shutdown-recovery-timeout 300
Router(config-l2vpn-bg-bd-mac-sec)#exit
Router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0.3
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#evi 100
Router(config-l2vpn-bg-bd-evi)#commit

/* Configure move count and move-interval */
Router#configure
Router(config)#evpn
Router(config-evpn)#mac secure
Router(config-evpn-mac-secure)#move-count 7
Router(config-evpn-mac-secure)#move-interval 30
Router(config-evpn-mac-secure)#commit
```

Running Configuration

This section shows the MAC loop prevention running configuration.

```
/* MAC Loop Prevention for VPLS */
l2vpn
 bridge group BG1
  bridge-domain BD1
   mac
    secure
     action shutdown
     threshold
     shutdown-recovery-timeout 300
    !
   interface GigabitEthernet0/2/0/0.1
   interface GigabitEthernet0/2/0/0.2
  !
```

```
/* MAC Loop Prevention for PW */
l2vpn
 bridge group BG1
  bridge-domain BD1
   mac
    secure
     action shutdown
     threshold
     shutdown-recovery-timeout 300
    !
   interface GigabitEthernet0/2/0/0.3
  !
   vfi VFI1
    neighbor 192.168.0.4 pw-id 3

/* MAC Loop Prevention for EVPN */
l2vpn
 bridge group BG1
  bridge-domain BD1
   mac
    secure
     action shutdown
     threshold
     shutdown-recovery-timeout 300
    !
   interface GigabitEthernet0/2/0/0.3
  !
   evi 100
!
/* Configure move-count and move-interval */
evpn
 mac
  secure
   move-count 7
   move-interval 30
 !
```

Verification

Verify that you have successfully configured the MAC Loop Prevention feature. The following show output displays the MAC security information:

```
Router# show l2vpn bridge-doamin detail
Bridge group: bg1, bridge-domain: bd1, id: 0, state: up, ShgId: 0, MSTi: 0
  MAC Secure: enabled, Logging: disabled, Action: shutdown, Threshold: enabled
MAC Secure Shutdown recovery timer : 300
List of ACs:
    AC: interface GigabitEthernet0/2/0/0.1, state is up
    AC: interface GigabitEthernet0/2/0/0.2, state is up
      MAC Secure: enabled, Logging: disabled, Action: shutdown, Threshold: enabled
      MAC Secure Shutdown recovery timer: 300
```

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 172.16.0.1 pw-id 1
     neighbor 192.168.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 10.0.0.1 255.0.0.0
```

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 10.0.0.1 pw-id 1
     neighbor 192.168.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 172.16.0.1 255.240.0.0
```

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 10.0.0.1 pw-id 1
     neighbor 172.16.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 192.168.0.1 255.255.0.0
```

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 10.0.0.1, 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 Cisco NCS 5500 Series Routers 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 Cisco NCS 5500 Series Routers 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 Cisco NCS 5500 Series Router 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.

GTP Load Balancing

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
GTP Load Balancing	Release 7.3.2	In addition to the source IP address, destination IP address, and port number, this functionality enables using the unique tunnel endpoint identifier (TEID) to compute load balancing (or hashing) of traffic in tunnels between ports. Using the TEID ensures that load balancing occurs even if the other parameters don’t have unique values, thus enabling efficient use of bandwidth and providing a reliable network. This functionality introduces the **hw-module loadbalancing gtp enable** command.

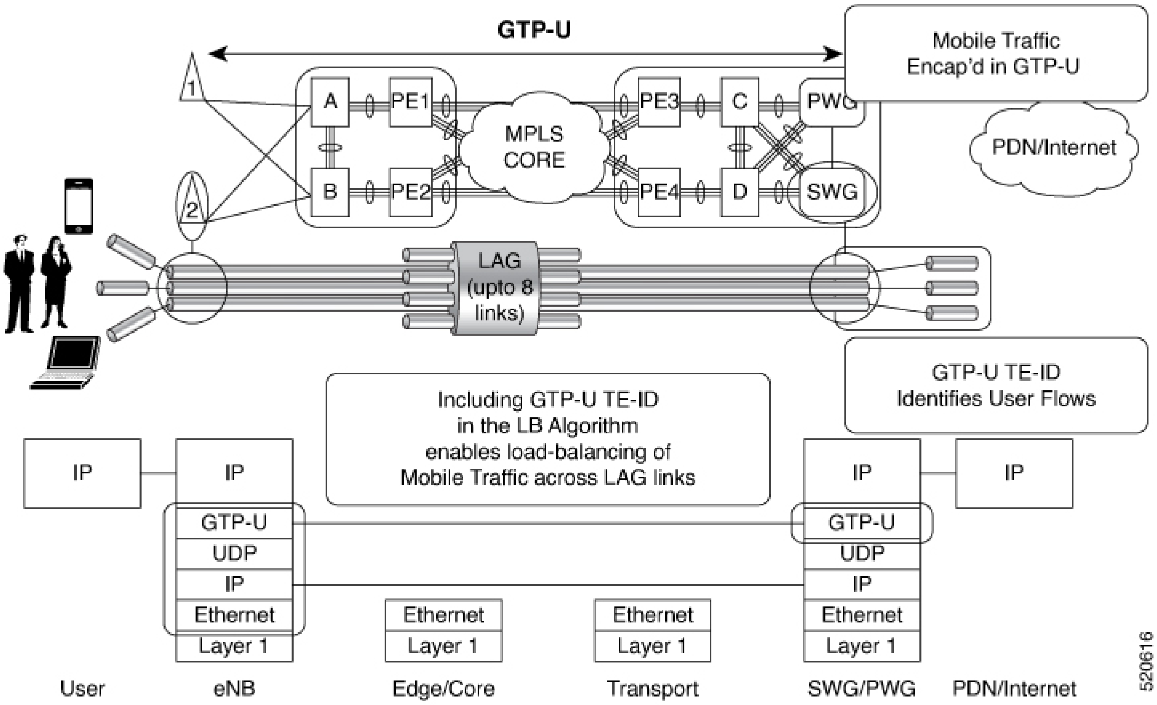
What is GTP?

GTP is a tunnel control and management protocol among General Packet Radio Service (GPRS) support nodes. Wireless networks use GTP tunnels to deliver mobile data. GTP includes GTP signaling (GTP-C) and data transfer (GTP-U) procedures. GTP-C specifies a tunnel control and management protocol, and is used to create, delete, and modify tunnels. GTP-U uses a tunneling mechanism to provide a service for carrying user data packets over the network.

What is GTP Load Balancing?

The following figure shows an illustration of the mobile transport GTP-U load balancing.

Figure 4: Mobile Transport GTP-U Load-Balancing



The global L3 flow-based load balancing considers the following fields:

- source address
- destination address
- router ID
- source port
- destination port

However, for GTP traffic, there are a limited number of unique values for these fields; this causes an uneven distribution of traffic. Sometimes, to facilitate redundancy and load balancing in a network, equal-cost paths exist to different destinations. Load balancing doesn't occur in such scenarios as the source and destination IP addresses and L4 ports are the same.

To achieve a greater distribution of traffic over equal-cost links, you can enable GTP TEID (Tunnel Endpoint ID) in the hash computation algorithm using the **hw-module loadbalancing gtp enable** command. Doing so ensures that the load balancing (hashing) computation algorithm includes the GTP TEID, unique for each traffic flow. The GTP load-balancing feature allows efficient distribution of traffic in mobile networks and provides increased reliability and availability for the network.

If the packet is UDP and the destination port is the GTP-U port (port number 2152), the GTP TEID is considered for loadbalancing.

The TEID in the GTP header of a GTP packet identifies individual tunnel endpoints, thus achieving better mobile traffic load balancing within any given GRE tunnel. Also, this helps in load-balancing GTP traffic over bundles at transit routers.

If TEID is present, load balancing based on tunnel endpoints is supported for Version 1 GTP packet and GTP version 2. For GTP version 0, load balancing occurs only if the fields described earlier have unique values, because there’s no TEID in version 0.

Note GTP load balancing is performed only for GTP-U (user data) packets. The GTP-C (control data) packets use a different destination port number of 2123 and hence, are subject to only the global L3 flow-based load balancing.

Note

- To enable hashing based on the tunnel id in GTP-U packets for ECMP and bundle member selection, use the **hw-module profile load-balance algorithm gtp** command.
- To enable hashing based on the tunnel id in GTP-U packets intead of Layer 4 packets when underlay network is MPLS, use the **hw-module profile load-balance algorithm gtp-mpls** command.

GTP Load Balancing Guidelines and Limitations

- GTP load balancing is performed on IPv4 or IPv6 incoming packets with GTP payloads and on MPLS incoming labeled packets.
- This feature supports GTP hashing only when the GTP UDP port is 2152.
- The number of MPLS label stacks in the transport layer is limited to three for GTP hashing. GTP hashing isn’t considered when the MPLS label stack exceeds three.
- You need not reboot the router after configuring or unconfiguring the **hw-module loadbalancing gtp enable** for GTP load balancing over MPLS to take effect.

Configuration

To enable GTP load balancing, configure the **hw-module loadbalancing gtp enable** command.

```
Router#config
Router(config)#hw-module loadbalancing gtp enable
Router(config)#commit
Router(config)#end
```

Verification

Run the following command to verify that GTP load balancing is enabled.

```
Router#show prm drv diagshell "getreg ING_GTP_CONTROL"
ING_GTP_CONTROL.ipipe0[1][0x3a004f00]=1: <GTP_HDR_FIRST_BYTE_MASK=0,
   GTP_HDR_FIRST_BYTE=0,GTP_ENABLE=1>
```

Flow Aware Transport Pseudowire (FAT PW)

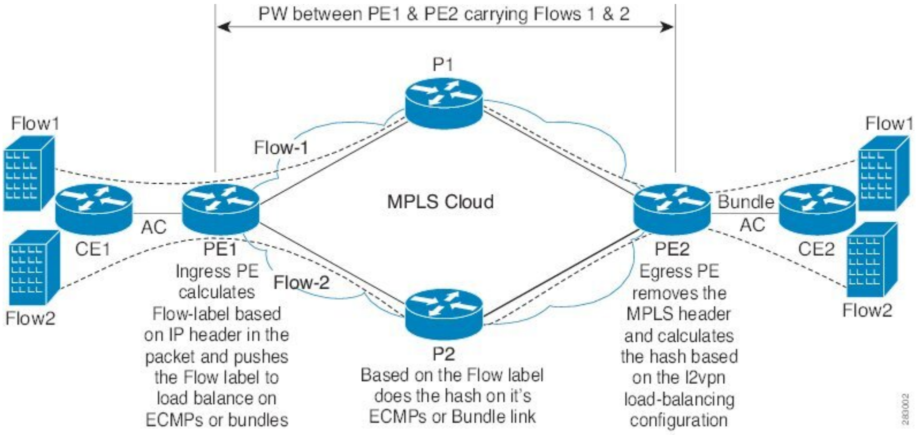
Routers typically loadbalance traffic based on the lower most label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric loadbalancing. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The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

Flow-Aware Transport Pseudowires (FAT PW) provide the capability to identify individual flows within a pseudowire and provide routers the ability to use these flows to loadbalance traffic. FAT PWs are used to loadbalance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on indivisible packet flows entering a pseudowire; and is inserted as the lower most label in the packet. Routers can use the flow label for loadbalancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.

Note Based on the number of traffic flows, a SMAC based hashing tuple creates the same flow label or less number of flow labels.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

Figure 5: FAT PW with two flows distributing over ECMPs and Bundle-Links


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

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

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it’s possible to distribute flows over ECMPs and link bundles.

You cannot send MPLS OAM ping traffic over a FAT PW, since there is no flow label support for MPLS OAM.

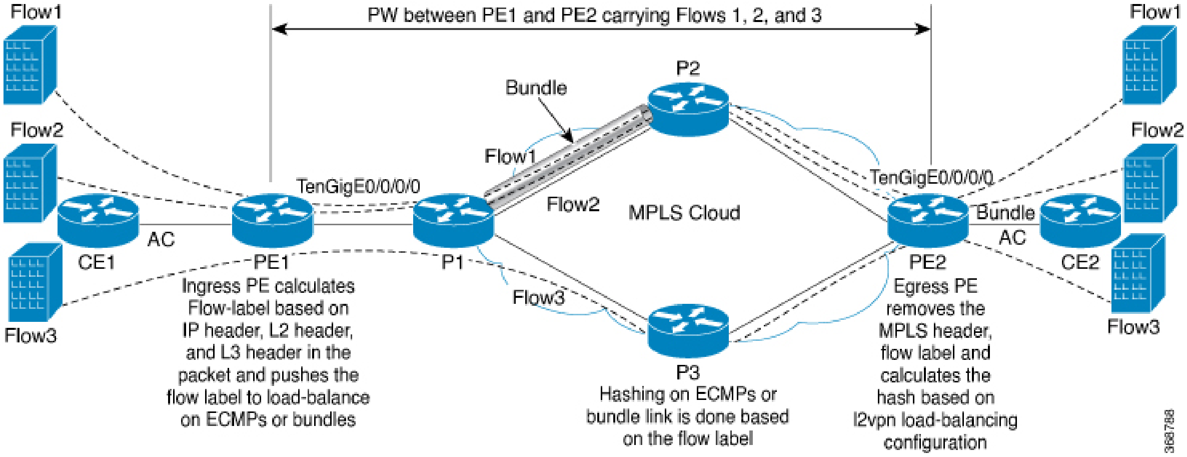
LDP-Based VPLS and VPWS FAT Pseudowire

The LDP-based VPLS and VPWS FAT Pseudowire feature enables provider (P) routers to use the flow-based load balancing to forward traffic between the provider edge (PE) devices. This feature uses Flow-Aware Transport (FAT) of pseudowires (PW) over an MPLS packet switched network for load-balancing traffic across LDP-based signaled pseudowires for Virtual Private LAN Services (VPLS) and Virtual Private Wire Service (VPWS).

FAT PWs provide the capability to identify individual flows within a PW and provide routers the ability to use these flows to load-balance the traffic. FAT PWs are used to load balance the traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on indivisible packet flows entering an imposition PE. This flow label is inserted as the lower most label in the packet. P routers use the flow label for load balancing to provide better traffic distribution across ECMP paths or link-bundled paths in the core. A flow is identified either by the source and destination IP address of the traffic, or the source and destination MAC address of the traffic.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

Figure 6: FAT PW with Two Flows Distributing over ECMPs and Bundle


Flow1
PW between PE1 and PE2 carrying Flows 1, 2, and 3
Flow1
P2
Bundle
Flow2
Flow1
Flow2
TenGigE0/0/0/0
Flow2
MPLS Cloud
TenGigE0/0/0/0
Bundle
AC
AC
CE1
PE1
P1
PE2
CE2
Flow3
Flow3
Flow3
Ingress PE calculates Flow-label based on IP header, L2 header, and L3 header in the packet and pushes the flow label to load-balance on ECMPs or bundles
P3
Hashing on ECMPs or bundle link is done based on the flow label
Egress PE removes the MPLS header, flow label and calculates the hash based on l2vpn load-balancing configuration
368788

An extra label is added to the stack, called the flow label, which is generated for each unique incoming flow on the PE. A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set. The flow label is inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it’s possible to distribute flows over ECMPs and link bundles.

In this topology, the imposition router, PE1, adds a flow label in the traffic. The disposition router, PE2, allows mixed types of traffic of which some have flow label, others do not. The P router uses flow label to load balance the traffic between the PEs. PE2 ignores the flow label in traffic, and uses one label for all unicast traffic.

Note When a FAT label is removed at PE2, load balancing is not performed based on the FAT label.

Configure LDP-Based VPLS and VPWS FAT Pseudowire

This feature is not supported for traffic across BGP-signaled pseudowires for VPLS and VPWS services.

Configuration Example

Perform this task to configure VPLS and VPWS FAT Pseudowire on both PE1 and PE2.

```
/* Configure LDP-based VPLS FAT Pseudowire */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class vpls
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# load-balancing
Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
Router(config-l2vpn-pwc-mpls-load-bal)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg0
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/5.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# vfi 2001
Router(config-l2vpn-bg-bd-vfi)# neighbor 192.0.2.1 pw-id 1
Router(config-l2vpn-bg-bd-vfi-pw)# pw-class vpls
Router(config-l2vpn-bg-bd-vfi-pw)# commit

/* Configure LDP-based VPWS FAT Pseudowire */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class vpws
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# load-balancing
Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
Router(config-l2vpn-pwc-mpls-load-bal)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group vpws
Router(config-l2vpn-xc)# p2p 1001
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/5.1001
Router(config-l2vpn-xc-p2p)# neighbor ipv4 192.0.2.1 pw-id 1001
Router(config-l2vpn-xc-p2p-pw)# pw-class vpws
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the running configuration of VPLS and VPWS FAT Pseudowire.

```
/* Configure LDP-based VPLS FAT Pseudowire */
l2vpn
pw-class vpls
  encapsulation mpls
   load-balancing
    flow-label both
   !
  !
bridge group bg0
  bridge-domain bd1
   interface TenGigE0/0/0/5.1
   !
   vfi 2001
    neighbor 192.0.2.1 pw-id 1
     pw-class vpls
    !
    !
```

```
/* Configure LDP-based VPWS FAT Pseudowire */
l2vpn
pw-class vpws
  encapsulation mpls
   load-balancing
    flow-label both
   !
  !
!
l2vpn
 xconnect group vpws
  p2p 1001
   interface interface TenGigE0/0/0/5.1001
   neighbor ipv4 192.0.2.1 pw-id 1001
    pw-class vpws
   !
  !
```

Verification

Verify that you have successfully configure the LDP-based VPLS and VPWS FAT Pseudowire feature.

```
/* Verify the LDP-based VPLS FAT Pseudowire configuration */
Router# show l2vpn bridge-domain group bg0 bd-name bd1 detail
Fri May 17 06:00:45.745 UTC
  List of VFIs:
    VFI 1 (up)
      PW: neighbor 192.0.2.1, PW ID 1, state is up ( established )
        PW class vpws, XC ID 0xc0000001
        Encapsulation MPLS, protocol LDP
        Source address 192.0.2.5
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set
        LSP : Up
        Flow Label flags configured (Tx=1,Rx=1), negotiated (Tx=1,Rx=1)

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ -------------------------
          Label        24000                          24000
          Group ID     0x0                            0x0
          Interface    1                              1
          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: 3221225473
        Create time: 12/05/2019 11:17:59 (4d18h ago)
        Last time status changed: 12/05/2019 11:24:03 (4d18h ago)
        MAC withdraw messages: sent 7, received 9
        Forward-class: 0
        Static MAC addresses:
        Statistics:
          packets: received 0 (unicast 0), sent 0
          bytes: received 0 (unicast 0), sent 0
          MAC move: 0
        Storm control drop counters:
          packets: broadcast 0, multicast 0, unknown unicast 0
          bytes: broadcast 0, multicast 0, unknown unicast 0
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 900 s, Type: inactivity
      MAC limit: 32000, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none

/* Verify the LDP-based VPWS FAT Pseudowire configuration */
Router# show l2vpn xconnect group vpws detail
Group vpws, XC 1001, state is up; Interworking none
  AC: TenGigE0/0/0/5.1001, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1001, 1001]
    MTU 1504; XC ID 0x47f; interworking none
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 192.0.2.1, PW ID 1001, state is up ( established )
    PW class vpws, XC ID 0xc0000548
    Encapsulation MPLS, protocol LDP
    Source address 192.0.2.2
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    LSP : Up
    Flow Label flags configured (Tx=1,Rx=1), negotiated (Tx=1,Rx=1)

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        25011                          25010
      Group ID     0xf000190                      0x228
      Interface    TenGigE0/0/0/5.1001            HundredGigE0/0/1/0.1001
      MTU          1504                           1504
      Control word disabled                       disabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x6                            0x6
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ -----------------------------
    Incoming Status (PW Status TLV):
```

```
    Status code: 0x0 (Up) in Notification message
  Outgoing Status (PW Status TLV):
    Status code: 0x0 (Up) in Notification message
  MIB cpwVcIndex: 3221226824
  Create time: 17/05/2019 05:52:59 (00:05:22 ago)
  Last time status changed: 17/05/2019 05:53:11 (00:05:10 ago)
  Statistics:
    packets: received 0, sent 0
   bytes: received 0, sent 0
```

Related Topics

- LDP-Based VPLS and VPWS FAT Pseudowire, on page 66

Associated Commands

- show l2vpn xconnect detail

PPPoE Traffic-Based Load Balancing

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
PPPoE Traffic-Based Load Balance using Flow-Aware Transport Labels	Release 7.4.1	This feature allows you to load balance the incoming PPPoE traffic received based on the inner PPPoE payload, source and destination IPv4 or IPv6 header. When you enable this feature, the router generates a unique Flow-Aware Transport (FAT) label for the incoming traffic based on inner IPv4 or IPv6 headers and uses the FAT labels for load balancing the PPPoE traffic. This feature introduces the **hw-module profile load-balance algorithm PPPoE** command.

Point-To-Point Protocol over Ethernet (PPPoE) is a network protocol that encapsulates Point-to-Point Protocol (PPP) frames inside Ethernet frames to allow data communication between two network entities or points.

For most of L2 switched packets, the source and destination MAC address in outer layer header remains the same, it becomes challenging to load balance the incoming PPPoE traffic based on outer headers. So, inner payloads or headers has been taken for hashing.

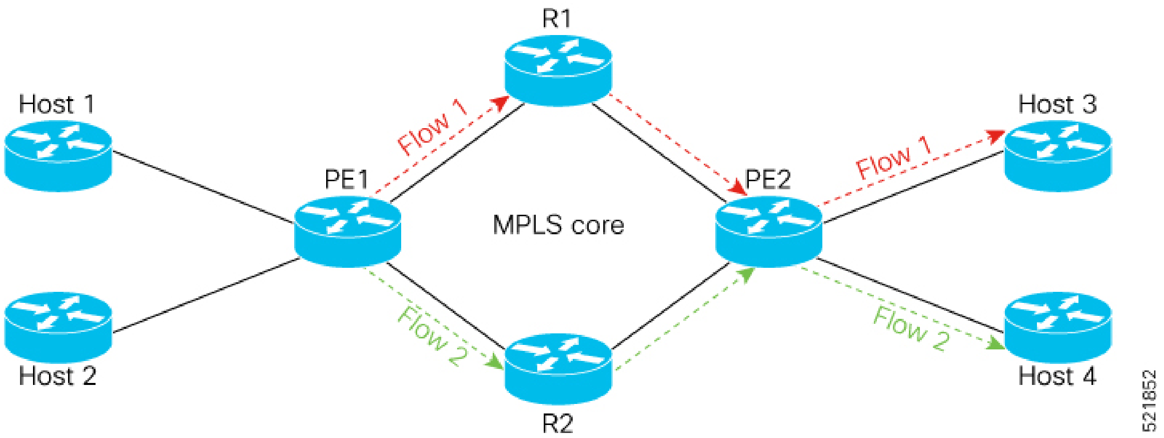
To enable PPPoE load-balancing feature, use the **hw-module profile load-balance algorithm PPPoE** command.

When you enable the feature, a unique Flow-Aware Transport (FAT) label is generated based on the inner PPPoE payload IPv4/IPv6 header. The traffic is load balanced using the FAT label.

A flow is defined as a sequence of related packets having the same source and destination pair which is sent from a source PE to a destination PE. FAT labels provide the capability to identify individual flows within a PW and provide routers the ability to use these flows to load balance the traffic.

For more information on Flow-Aware Transport, see LDP-Based VPLS and VPWS FAT Pseudowire, on page 66.

Topology



In this topology, PE1 receives two packets flows, let’s consider Flow 1 as traffic from Host 1 (source) to Host 3 (destination) and Flow 2 as traffic from Host 2 (source) to Host 4 (destination).

- **Flow 1:** When PE1 receives traffic from Host 1, PE1 adds a unique FAT label for flow 1 and forwards the traffic through R1. So when ever Host 1 sends the traffic to Host 3, PE1 always forwards the traffic through R1.
- **Flow 2:** When PE1 receives traffic from Host 2, it checks for the destination address and adds a different FAT label for Flow 2. PE1 forwards the traffic through R2. So when ever Host 2 sends the traffic to Host 4, PE1 always forwards the traffic through R2.

Restrictions

- Supports only transit PPPoE header over Ethernet transport.
- The PPPoE load-balancing feature can’t coexist with features such as SRv6 and GUE.

Configure Load balancing for PPPoE Traffic

To enable load balancing for PPPoE traffic:

Prerequisites

You must configure the FAT label for L2VPN, before you enable the PPPoE load-balancing feature. For configuring the FAT label, see LDP-Based VPLS and VPWS FAT Pseudowire, on page 66.

Configuration Example

Note After you enable or disable the PPPoE load-balancing feature, you should reload the line card for this configuration to take effect.

```
Router# configure terminal
Router(config)# hw-module profile load-balance algorithm pppoe
Router(config)# commit
Router(config)# exit
Router# reload
```

To disable load balancing for PPPoE traffic:

```
Router# configure terminal
Router(config)# no hw-module profile load-balance algorithm pppoe
Router(config)# commit
Router(config)# exit
Router# reload
```

Verification

```
Router# Show interface accounting
Bundle-Ether1.1
Protocol Pkts In Chars In Pkts Out Chars Out
IPV4_UNICAST 4494 251664 4494 251664
IPV6_UNICAST 3 228 0 0
MPLS 0 0 252573 35360220
IPV6_ND 12 1032 4 320
CLNS 152 91885 26 2028

Bundle-Ether2.1
Protocol Pkts In Chars In Pkts Out Chars Out
IPV4_UNICAST 4494 251664 4494 251664
IPV6_UNICAST 3 228 0 0
MPLS 0 0 252573 35360220
IPV6_ND 12 1032 4 320
CLNS 155 95324 26 2028
```

Static L2VPN P2P over MPLS over GRE Tunnel

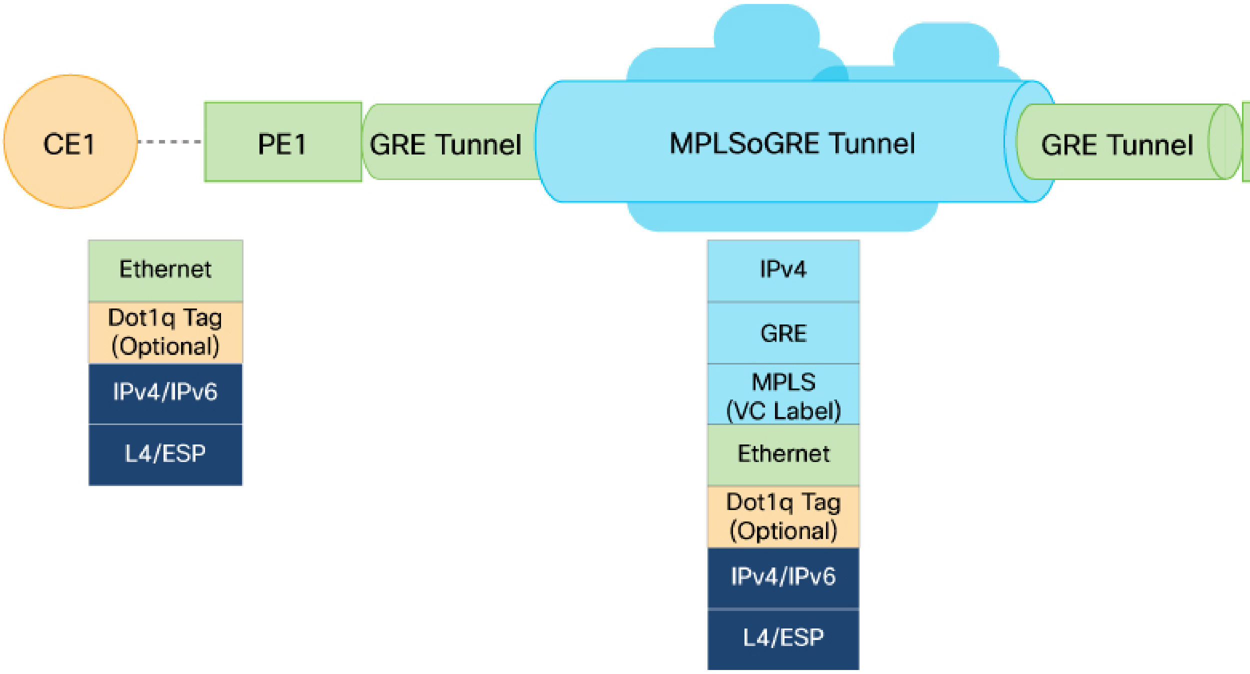
Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Static L2VPN P2P Configuration over MPLS over Single-Pass GRE Tunnel	Release 7.10.1	Introduced in this release on: NCS 5500 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards) We help you achieve the designated line rate for your network by enabling a single-pass GRE for static L2VPN point-to-point MPLS over GRE traffic. Earlier, you could transport such traffic using two-pass GRE tunnels, which used the recycle functonality to ensure payload transportation while optimizing the line rate.
Static L2VPN P2P over MPLS over GRE Tunnel	Release 7.5.4	You can now transport any frames over L2VPN P2P over MPLSoGRE network to get L2 PW services. MPLSoGRE tunnels MPLS packets encapsulated with GRE header over a non-MPLS network. Earlier you could transport the traffic using only MPLS tunnels.

The Multiprotocol Label Switching (MPLS) over GRE (MPLSoGRE) functionality provides a mechanism for tunneling MPLS packets over a non-MPLS network. MPLSoGRE is used to encapsulate MPLS packets inside IP tunnels.

Figure 7: Static L2VPN P2P over MPLS over GRE Tunnel

The following figure depicts the L2VPN and GRE tunnel connections at PE1 and PE2 routers to send MPLS traffic over GRE:



1. When a tagged or untagged Ethernet frame is sent from a CE1 router to a PE1 router, the PE1 router adds the MPLS label to the packet for each customer as per VLAN tag.

2. After attaching the MPLS label, the PE1 router encapsulates the packet with a GRE header and forwards it to a GRE tunnel connected to it.

   The encapsulation process comprises the following actions at the PE1 router in single-pass:

   - VC label is attached, and GRE header is encapsulated.
   - Destination interface is checked and the packet is forwarded to the GRE tunnel.

   The encapsulation process comprises the following actions at the PE1 router in two-pass:

   - First-pass: VC label is attached, and GRE header is encapsulated.
   - Second-pass: Destination interface is checked and the packet is forwarded to the GRE tunnel.

3. GRE tunnel forwards the packets to the PE2 router.

4. When the PE2 router receives the GRE encapsulated packets, it decapsulates the GRE information, and forwards the packet to the CE2 router based on the MAC address or VLAN, or both. The entire process of decapsulation is performed in single-pass or two-pass GRE.

   The decapsulation process comprises the following actions at the PE2 router in single-pass:

   - GRE header and VC label are decapsulated.
   - Packet is forwarded to the CE2 router.

   The decapsulation process comprises the following actions at the PE2 router in two-pass:

   - First-pass: GRE header is decapsulated.
   - Second-pass: VC label is terminated and the packet is forwarded to the CE2 router.

Limitations and Guidelines

Limitations

Single-pass GRE Static L2VPN P2P over MPLS over GRE Tunnel supports:

- 1000 GRE tunnels for each PW
- 2000 L2 subinterfaces
- 2000 static PW configurations
- 2000 L2 ACs

Two-pass GRE Static L2VPN P2P over MPLS over GRE Tunnel supports:

- 8000 L2VPN PW tunnels
- 16 GRE tunnels for each PW
- 8000 L2VPN PW tunnels for each PE-pair
- 8000 L2 subinterfaces
- Throughput is reduced since its two-pass GRE process.

Guidelines

L2 interfaces of CE1 and CE2 routers in single-pass GRE static L2VPN P2P over MPLS over GRE tunnel supports:

- Minimum two and maximum eight member links.
- Per VLAN or per MPLS Lable, or both per VLAN and per MPLS label services for L3, L2VPN, mcast flow.
    - L2 and L3 traffic can be on different VLAN subinterfaces.
    - L3 and mcast traffic can be on the same VLAN subinterface.
- Load-balancing for the decap traffic is done on your L2 header (SMAC, DMAC).

L2 interfaces of CE1 and CE2 routers in two-pass GRE static L2VPN P2P over MPLS over GRE tunnel supports:

- Maximum of eight member links.
- Subinterface configuration.
- An unmatched VLAN association to provide VLAN bundling.
- L2 and L3 traffic together on the same port or physical or interface.

Configure Static L2VPN P2P over MPLS over GRE Tunnel

Configuring single-pass or two-pass GRE static L2VPN P2P over MPLS over GRE Tunnel involves:

- Configuring Static Point-to-Point Cross-Connect
- Configuring GRE Tunnel

Configuring Static Point-to-Point Cross-Connect

You must configure a static cross-connect circuits in a Layer 2 VPN to send traffic over MPLSoGRE in single-pass and two-pass GRE configuration.

Configuration Example at PE1

This example shows how to configure static cross-connect P2P circuit at edge router (PE1):

```
/* Neighbour-interface Configuration*/
Router#config
Router(config)#interface Loopback 1
Router(config-if)# ipv4 address 172.16.0.1 255.255.255.0
Router(config-subif)#commit
Router(config-subif)#exit

/*L2VPN and MPLS Configuration*/
Router#config
Router(config)# interface Loopback 501
Router(config-if)# ipv4 address 192.168.1.1 255.255.255.0
Router(config-subif)# l2vpn
Router(config-l2vpn)# xconnect group xc1
Router(config-l2vpn-xc)# p2p p2p1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether100.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.2 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 1000 remote 4000
Router(config-l2vpn-xc-p2p-pw)# mpls static address-family ipv4 unicast
Router(config-l2vpn-mpls-static-af)# lsp l2_gre_v4_1
Router(config-l2vpn-mpls-static-lsp)# in-label 24044 allocate per-prefix 172.16.0.2/32
Router(config-l2vpn-mpls-static-lsp)# forward path 1 nexthop tunnel-ip1 out-label pop
Router(config-l2vpn-mpls-static-lsp)# commit
Router(config-l2vpn-mpls-static-lsp)# exit
```

Running configuration

```
interface Loopback1
 ipv4 address 172.16.0.1 255.255.255.0
!
interface Loopback501
 ipv4 address 192.168.1.1 255.255.255.0
!


l2vpn
 xconnect group xc1
  p2p p2p1
   interface Bundle-Ether100.1
   neighbor ipv4 172.16.0.2 pw-id 1
    mpls static label local 1000 remote 4000


mpls static
 address-family ipv4 unicast
 !
 lsp l2_gre_v4_1
```

```
  in-label 24044 allocate per-prefix 172.16.0.2/32
  forward path 1 nexthop tunnel-ip1 out-label pop
```

Configuration Example at PE2

This example shows how to configure static cross-connect p2p circuit at edge router (PE2):

```
/* Neighbour-interface Configuration*/
Router#config
Router(config)#interface Loopback 1
Router(config-if)# ipv4 address 172.16.0.2 255.255.255.0
Router(config-subif)#commit
Router(config-subif)#exit

/*L2VPN and MPLS Configuration*/
Router# config
Router(config)# interface Loopback 501
Router(config-if)# ipv4 address 192.168.2.1 255.255.255.255
Router(config-subif)# l2vpn
Router(config-l2vpn)# xconnect group xc-1
Router(config-l2vpn-xc)# p2p p2p1
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether200.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 4000 remote 1000
Router(config-l2vpn-xc-p2p-pw)# mpls static address-family ipv4 unicast
Router(config-l2vpn-mpls-static-af)# lsp l2_gre_v4_1
Router(config-l2vpn-mpls-static-lsp)# in-label 24011 allocate per-prefix 172.16.0.1/32
Router(config-l2vpn-mpls-static-lsp)# forward path 1 nexthop tunnel-ip1 out-label pop
Router(config-l2vpn-mpls-static-lsp)# commit
Router(config-l2vpn-mpls-static-lsp)# exit
```

Running Configuration

```
interface Loopback1
 ipv4 address 172.16.0.2 255.255.255.0
!


interface Loopback501
 ipv4 address 192.168.2.1 255.255.255.255
!

l2vpn
 xconnect group xc-1
  p2p p2p1
   interface Bundle-Ether200.1
   neighbor ipv4 172.16.0.1 pw-id 1
    mpls static label local 4000 remote 1000
   !



mpls static
 address-family ipv4 unicast
 !
 lsp l2_gre_v4_1
  in-label 24011 allocate per-prefix 172.16.0.1/32
  forward path 1 nexthop tunnel-ip1 out-label pop
 !
```

Configuring GRE Tunnel Interface

You must configure a GRE tunnel to tunnel the traffic over MPLSoGRE.

To configure a single-pass GRE, see *Configuring GRE Tunnels* chapter from *Interface and Hardware Component Configuration Guide for Cisco NCS 5500 Series Routers*

To configure a two-pass GRE, use the following configuration.

Configuration Example at PE1

This example shows how to configure a GRE tunnel at the edge router (PE1):

```
Router# config
Router(config)# interface tunnel-ip 1
Routers(config-if)# ipv4 address 178.16.0.1 255.255.255.0
Router(config-if)# tunnel mode gre ipv4
Router(config-if)# tunnel source 192.168.1.1
Router(config-if)# tunnel destination 192.168.2.1
Router(config-if)# commit
```

Running Configuration

```
interface tunnel-ip1
 ipv4 address 178.16.0.1 255.255.255.0
 tunnel mode gre ipv4
 tunnel source 192.168.1.1
 tunnel destination 192.168.2.1
!
```

Configuration Example at PE2

This example shows how to configure a GRE tunnel at the edge router (PE2):

```
Router(config)# interface tunnel-ip1
Router(config-if)# ipv4 address 178.16.0.2 255.255.255.0
Router(config-if)# tunnel mode gre ipv4
Router(config-if)# tunnel source 192.168.2.1
Router(config-if)# tunnel destination 192.168.1.1
Router(config-if)# commit
```

Running Configuration

```
interface tunnel-ip1
 ipv4 address 178.16.0.2 255.255.255.0
 tunnel mode gre ipv4
 tunnel source 192.168.2.1
 tunnel destination 192.168.1.1
!
```

Verification

You can verify the loopback configuration with swaped local and remote MPLS label, or tunnel source and destination IP addresses for which MPLS traffic is looped back with GRE tunnel to the source router.

Verification at PE1

```
/*Verification of Single-Pass GRE Configuration*/
Router# show l2vpn xconnect group xc1 detail
Group xc1, XC p2p1, state is up; Interworking none
  AC: Bundle-Ether100.1, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1504; XC ID 0xa0000001; interworking none
    Statistics:
      packets: received 122159214, sent 18454816
      bytes: received 15147742536, sent 2288397184
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 172.16.0.2, PW ID 1, state is up
    PW class not set, XC ID 0xc0000001
    Encapsulation MPLS, protocol none
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Ignore MTU mismatch: Disabled
    Transmit MTU zero: Disabled
    LSP : Up
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        1000                           4000
      Interface    Bundle-Ether100.1
      MTU          1504
      Control word disabled
      PW type      Ethernet
      VCCV CV type 0x2
                   (LSP ping verification)
      VCCV CC type 0x6
                   (router alert label)
                   (TTL expiry)
      ------------ ------------------------------ -----------------------------
    MIB cpwVcIndex: 3221225473
    Create time: 02/12/2022 16:52:53 (01:25:43 ago)
    Last time status changed: 02/12/2022 16:55:39 (01:22:58 ago)
    Statistics:
      packets: received 18454816, sent 122159214
      bytes: received 2288397184, sent 15147742536
```

```
/*Verification of Two-Pass GRE Configuration*/
Router# show interfaces tunnel-ip 1
tunnel-ip1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is Tunnel
  Internet address is 178.16.0.1/24
  MTU 1500 bytes, BW 100 Kbit (Max: 100 Kbit)
     reliability 255/255, txload 255/255, rxload 255/255
  Encapsulation TUNNEL_IP,  loopback not set,
  Last link flapped 01:23:27
  Tunnel TOS 0
  Tunnel mode GRE IPV4
  Keepalive is disabled.
  Tunnel source 192.168.1.1, destination 192.168.2.1/32
  Tunnel TTL 255
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 00:01:08
  5 minute input rate 121115000 bits/sec, 114693 packets/sec
  5 minute output rate 958179000 bits/sec, 767771 packets/sec
     57800516 packets input, 7629668112 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     385336514 packets output, 60112496184 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
```

Verification at PE2

```
/*Verification of Single-Pass GRE Configuration*/
Router# show l2vpn xconnect group xc-1 detail
Group xc-1, XC p2p1, state is up; Interworking none
```

```
  AC: Bundle-Ether200.1, state is up
    Type VLAN; Num Ranges: 1
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1504; XC ID 0xa0000001; interworking none
    Statistics:
      packets: received 122159214, sent 18454816
      bytes: received 15147742536, sent 2288397184
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 172.16.0.1, PW ID 1, state is up
    PW class not set, XC ID 0xc0000001
    Encapsulation MPLS, protocol none
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Ignore MTU mismatch: Disabled
    Transmit MTU zero: Disabled
    LSP : Up
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        4000                           1000
      Interface    Bundle-Ether200.1
      MTU          1504
      Control word disabled
      PW type      Ethernet
      VCCV CV type 0x2
                   (LSP ping verification)
      VCCV CC type 0x6
                   (router alert label)
                   (TTL expiry)
      ------------ ------------------------------ -----------------------------
    MIB cpwVcIndex: 3221225473
    Create time: 02/12/2022 16:52:53 (01:25:43 ago)
    Last time status changed: 02/12/2022 16:55:39 (01:22:58 ago)
    Statistics:
      packets: received 18454816, sent 122159214
      bytes: received 2288397184, sent 15147742536

/*Verification of Two-Pass GRE Configuration*/
Router# show interfaces tunnel-ip 1
tunnel-ip1 is up, line protocol is up
  Interface state transitions: 1
  Hardware is Tunnel
  Internet address is 178.16.0.2/24
  MTU 1500 bytes, BW 100 Kbit (Max: 100 Kbit)
     reliability 255/255, txload 255/255, rxload 255/255
  Encapsulation TUNNEL_IP,  loopback not set,
  Last link flapped 01:23:27
  Tunnel TOS 0
  Tunnel mode GRE IPV4
  Keepalive is disabled.
  Tunnel source 192.168.2.1, destination 192.168.1.1/32
  Tunnel TTL 255
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters 00:01:08
  5 minute input rate 121115000 bits/sec, 114693 packets/sec
  5 minute output rate 958179000 bits/sec, 767771 packets/sec
     57800516 packets input, 7629668112 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     385336514 packets output, 60112496184 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
```

Dynamic Address Resolution Protocol (ARP) Inspection

Table 12: Feature History Table

Feature Name	Release Information	Description
Dynamic Address Resolution Protocol (ARP) Inspection (DAI)	Release 7.9.1	The routers can now determine the validity of an Address Resolution Protocol (ARP) packet based on valid MAC address to IP address bindings stored in a trusted database built at runtime by DHCP snooping. With this feature, the router relays only the valid ARP requests and responses, thus preventing the ARP poisoning attacks. This feature introduces the following: • **CLI:** New **dynamic-arp-inspection** command. • **Yang Data Model**: Cisco-IOS-XR-l2vpn-oper.yang and Cisco-IOS-XR-ipv4-arp-oper.yang (see GitHub, YANG Data Models Navigator)

Dynamic ARP Inspection (DAI) prevents the Address Resolution Protocol (ARP) poisoning attacks by intercepting all ARP requests and responses. Each traffic packet is verified for valid MAC address to IP address bindings before the local ARP cache is updated or when the traffic packet is sent to the appropriate destination. The router drops the invalid ARP packets. The DAI determines the validity of an ARP packet based on valid MAC address, Source IP address, and Source Interface information available in the DHCP snooping entries available in the router.. In addition, DAI can also validate ARP packets against user-configured ARP ACLs to handle hosts that use statically configured IP addresses. You cn configure the DAI to drop ARP packets when the IP addresses in the packet are invalid or when the MAC addresses in the body of the ARP packet don't match the addresses specified in the Ethernet header.

Prerequisites

- Enable DHCP snooping. For more information, see *DHCP Snooping* section in *Implementing the Dynamic Host Configuration Protocol* chapter in *IP Addresses and Services Configuration Guide for Cisco NCS 5500 Series Routers*.

To configure the Dynamic ARP Inspection, do the following:

Configuration

```
Router# configure
Enters the Global Configuration mode.

Router(config)# l2vpn
Enters the l2vpn configuration mode.

Router(config-l2vpn)# bridge group csco
Creates a bridge group so that it can contain bridge domains and then assigns network
interfaces to the bridge domain.

Router(config-l2vpn-bg)# bridge-domain abc
Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Router(config-l2vpn-bg-bd)# dynamic-arp-inspection logging
Configures logging for dynamic ARP inspection configuration submode.

Router(config-l2vpn-bg-bd)# dynamic-arp-inspection address-validation
Configures logging for dynamic ARP inspection configuration submode.

Router(config-l2vpn-bg-bd)# commit
Router(config-l2vpn-bg-bd)# exit
```

Running Configuration

```
Router(config)# show runnig-config l2vpn
l2vpn
  bridge group csco
    bridge-domain abc
       dynamic-arp-inspection logging
       dynamic-arp-inspection address-validation
       !
    !
!
```

Verification

```
Router(config)# show l2vpn bridge-domain abc detail
Legend: pp = Partially Programmed.
Bridge group: evpn-aa-irb-inter, bridge-domain: evpn, id: 1797, state: up, ShgId: 0, MSTi:
 0
  Coupled state: disabled
  VINE state: EVPN-IRB
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw for Access PW: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 64000, Action: none, Notification: syslog
  MAC limit reached: no, threshold: 99%
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  Dynamic ARP Inspection: enabled, Logging: enabled
  IP Source Guard: disabled, Logging: disabled
  DHCPv4 Snooping: disabled
  DHCPv4 Snooping profile: none
  IGMP Snooping: disabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 1798
  Filter MAC addresses:
  P2MP PW: disabled
  Multicast Source: Not Set
  Create time: 11/12/2020 02:02:56 (04:55:20 ago)
  No status change since creation
  ACs: 2 (2 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
      evi: 2001
      XC ID 0x800006a7
     Statistics:
       packets: received 0 (unicast 0), sent 0
       bytes: received 0 (unicast 0), sent 0
       MAC move: 0
  List of ACs:
    AC: BVI10001, state is up
      Type Routed-Interface
      MTU 2000; XC ID 0x80000fa3; interworking none
      BVI MAC address:
        0088.0088.0088
      Split Horizon Group: Access
      PD System Data: AF-LIF-IPv4: 0x00000000  AF-LIF-IPv6: 0x00000000 FRR-LIF: 0x00000000

    AC: Bundle-Ether30001.2001, state is up
      Type VLAN; Num Ranges: 1
      Outer Tag: 3001
      Rewrite Tags: []
      VLAN ranges: [2001, 2001]
      MTU 1500; XC ID 0xa00005e0; interworking none; MSTi 1
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 64000, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 99%
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      E-Tree: Root
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
      Static MAC addresses:
      Statistics:
        packets: received 404672709 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 0
        bytes: received 30835628366 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 0
        MAC move: 0
      Storm control drop counters:
        packets: broadcast 0, multicast 0, unknown unicast 0
        bytes: broadcast 0, multicast 0, unknown unicast 0
      Dynamic ARP inspection drop counters:
        packets: 0, bytes: 0
      IP source guard drop counters:
        packets: 0, bytes: 0
      PD System Data: AF-LIF-IPv4: 0x00018919  AF-LIF-IPv6: 0x0001891a FRR-LIF: 0x00000000
```

```
List of Access PWs:
List of VFIs:
List of Access VFIs:
```

IP SourceGuard

Table 13: Feature History Table

Feature Name	Release Information	Description
IP SourceGuard (IPSG)	Release 7.9.1	You can now achieve source IP address filtering on a Layer 2 port, to prevent a malicious host from manipulating a legitimate host by assuming the legitimate IP address of the host. This feature uses dynamic DHCP snooping and static IP source binding to match IP addresses to hosts. This filtering limits the ability of a host to attack the network by claiming the IP address of a neighbor host. This feature introduces the following: • **CLI:** New **ip-source-guard** command. • **Yang Data Model**: Cisco-IOS-XR-l2vpn-oper.yang (see GitHub, YANG Data Models Navigator)

IP SourceGuard (IPSG) is a security feature that filters traffic based on the DHCP snooping binding database and on manually configured IP source bindings to restrict IP traffic on nonrouted Layer 2 interfaces.

Initially, the router blocks all IP traffic on the Ethernet Flow Points (EFPs), except for DHCP packets. After a client receives an IP address from the DHCP server, or after static IP source binding is configured by the administrator, all traffic with that IP source address is permitted from that client. Traffic from other hosts is denied.

Prerequisites

- Enable DHCP snooping. For more information, see *DHCP Snooping* section in *Implementing the Dynamic Host Configuration Protocol* chapter in *IP Addresses and Services Configuration Guide for Cisco NCS 5500 Series Routers*.

To configure the IP SourceGuard, do the following:

You can enable IP source guard under the bridge domain or the bridge port. By default, bridge ports under a bridge inherit the IP source guard configuration from the parent bridge. By default, IP source guard is disabled on the bridges.

Configuration

```
Router# configure

Router(config)# l2vpn

Creates a bridge group so that it can contain bridge domains and then assigns network
interfaces to the bridge domain.
Router(config-l2vpn)# bridge group csco

Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.
Router(config-l2vpn-bg)# bridge-domain abc

Enters the IP source guard configuration submode and enables source IP address filtering
on a Layer 2 port.
Router(config-l2vpn-bg-bd)# ip-source-guard logging

Router(config-l2vpn-bg-bd)# commit
Router(config-l2vpn-bg-bd)# exit
```

Running Configuration

```
Router(config)# show runnig-config l2vpn
l2vpn
  bridge group csco
    bridge-domain abc
       ip-source-guard logging
       !
    !
!
```

Verification

```
Router(config)# show l2vpn forwarding interface g0/0/0/1.10 detail location 0/0/CPU0
LC/0/0/CPU0:Jun 16 13:32:25.334 : l2fib[188]: %L2-L2FIB-5-SECURITY_IPSG_VIOLATION_AC : IP
source guard in AC  GigabitEthernet0_0_0_7.1001 detected violated packet - source MAC:
0000.0000.0200, destination MAC: 0000.0003.0000, source IP: 130.0.0.1, destination IP:
125.34.2.5
LC/0/5/CPU0:Jun 16 13:33:25.530 : l2fib[188]: %L2-L2FIB-5-SECURITY_IPSG_VIOLATION_AC : IP
source guard in AC Bundle-Ether100.100  detected violated packet - source MAC:
0000.0000.0064, destination MAC: 0000.0040.0000, source IP: 14.5.1.3, destination IP:
45.1.1.10
```