Configure Point-to-Point Layer 2 Services

Point-to-point service basically emulates a transport circuit between two end nodes so the end nodes appear to be directly connected over a point-to-point link. This can be used to connect two sites.

This section introduces you to point-to-point Layer 2 services, and also describes the configuration procedures to implement it.

The following point-to-point services are supported:

- Local Switching—A point-to-point internal circuit on a router, also known as local connect. Local switching allows switching of Layer 2 data between two attachment circuits on the same device.
- Attachment circuit—An attachment circuit (AC) is a physical or logical port or circuit that connects a CE device to a PE device.
- Pseudowires—A virtual point-to-point circuit from one PE router to another. Pseudowires are implemented over the MPLS network.

Note Point-to-point Layer 2 services are also called as MPLS Layer 2 VPNs.

- Pseudowire over MPLS , on page 1
- PW over MPLS Supported Modes, on page 5
- Pseudowire Headend, on page 15
- Preferred Tunnel Path, on page 36
- Configure Local Switching Between Attachment Circuits, on page 40
- MPLS PW Traffic Load Balancing on P Router, on page 42
- L2VPN Traffic Load Balancing on PE Router, on page 46
- G.8032 Ethernet Ring Protection Switching, on page 53

Pseudowire over MPLS

Table 1: Feature History Table

Feature Name	Release Information	Feature Description

Pseudowire over MPLS	Release 7.3.15	This feature allows you to tunnel two L2VPN Provider Edge (PE) devices to transport L2VPN traffic over an MPLS core network. MPLS labels are used to transport data over the pseudowire.

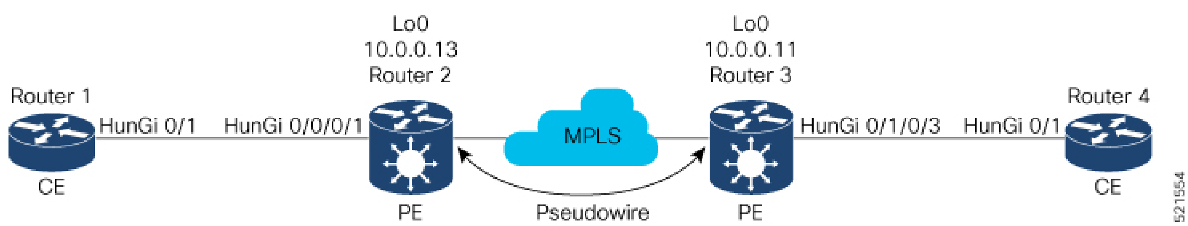
A pseudowire (PW) is a point-to-point connection between two provider edge (PE) devices which connects two attachment circuits (ACs). The two ACs connected at each PE are linked by a PW over the MPLS network, which is the MPLS PW.

PWs provide a common intermediate format to transport multiple types of network services over a Packet Switched Network (PSN) – a network that forwards packets – IPv4, IPv6, MPLS, Ethernet.

Pseudowire over MPLS or Ethernet-over-MPLS (EoMPLS) provides a tunneling mechanism for Ethernet traffic through an MPLS-enabled Layer 3 core network. PW over MPLS encapsulates Ethernet protocol data units (PDUs) using MPLS labels to forward them across the MPLS network.

Topology

Here is an example that showcases how the L2VPN traffic is transported using the PW over MPLS network.



- CEs are connected to PEs using the attachment circuit (AC).
- PW is configured on the PE devices to connect two PEs over an MPLS core network.

Consider a traffic flow from Router 1 to Router 4. Router 1 sends the traffic to Router 2 through the AC. Router 2 adds the MPLS PW label and sends it to Router 3 through the PW. Each PE needs to have an MPLS label in order to reach the loopback of the remote PE. This label, usually called the Interior Gateway Protocol (IGP) label, can be learned through the MPLS Label Distribution Protocol (LDP) or MPLS Traffic Engineering (TE).

One PE advertises the MPLS label to the other PE for PW identification. Router 3 identifies traffic with MPLS label and sends it to the AC connected to Router 4 after removing the MPLS label.

You can configure static or dynamic point-to-point connections.

Configure Static Point-to-Point Connections Using Cross-Connect Circuits

This section describes how you can configure static point-to-point cross connects in a Layer 2 VPN.

Requirements and Limitations

Before you can configure a cross-connect circuit in a Layer 2 VPN, ensure that the following requirements are met:

- The CE and PE routers are configured to operate in a network.

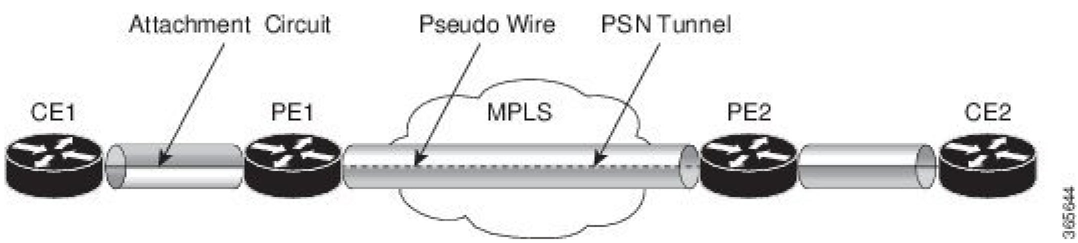
- The name of a cross-connect circuit is configured to identify a pair of PE routers and must be unique within the cross-connect group.
- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect circuit.
- A static virtual circuit local label is globally unique and can be used in only one pseudowire.

Note Static pseudowire connections do not use LDP for signaling.

Topology

The following topology is used to configure static cross-connect circuits in a Layer 2 VPN.

Figure 1: Static Cross-Connect Circuits in a Layer 2 VPN



Configuration

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigEt0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.3 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 50 remote 40
Router(config-l2vpn-xc-p2p-pw)# commit
```

```
/*Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/2/0/0.4
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.4 pw-id 100
Router(config-l2vpn-xc-p2p-pw)# mpls static label local 40 remote 50
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* On PE1 */
!
l2vpn
 xconnect group XCON1
  p2p xc1
   interface HundredGigE0/1/0/0.1
   neighbor ipv4 10.0.0.3 pw-id 100
    mpls static label local 50 remote 40
```

```
!

/* On PE2 */
!
l2vpn
 xconnect group XCON2
  p2p xc1
   interface HundredGigE0/2/0/0.4
   neighbor ipv4 10.0.0.4 pw-id 100
    mpls static label local 40 remote 50
  !
```

Verification

```
/* Verify the static cross connect on PE1 */
Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
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
------------------------   ----------------------------   ----------------------------
XCON1      xc1        UP   Hu0/1/0/0.1            UP      10.0.0.3  100    UP
----------------------------------------------------------------------------------------

/* Verify the static cross connect on PE2 */

Router# show l2vpn xconnect
Tue Apr 12 20:18:02.971 IST
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
------------------------   ----------------------------   ----------------------------
XCON2      xc1        UP   Hu0/2/0/0.4            UP      10.0.0.4  100    UP
----------------------------------------------------------------------------------------
```

Configure Dynamic Point-to-point Cross-Connects

Perform this task to configure dynamic point-to-point cross-connects.

Note For dynamic cross-connects, LDP must be up and running.

Configuration

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group vlan_grp_1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HunGigE 0/0/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
configure
  l2vpn
   xconnect group vlan_grp_1
    p2p vlan1
    interface HunGigE 0/0/0/0.1
    neighbor 10.0.0.1 pw-id 1
!
```

PW over MPLS Supported Modes

The PW over MPLS support these modes:

Ethernet Port Mode

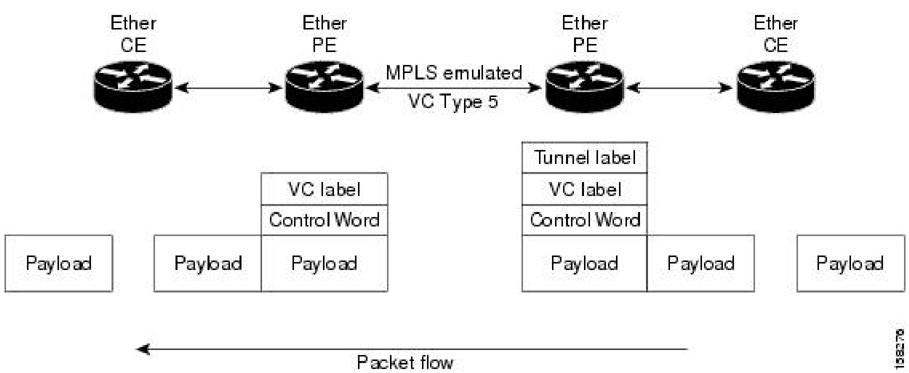
Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire VC Type 5	Release 7.3.15	With this feature, Ethernet port mode is supported for pseudowire over MPLS. The virtual connection (VC) type 5 is known as an Ethernet port-based PW. In this mode, both ends of a pseudowire are connected to Ethernet ports and allow a complete ethernet trunk to be transported. The ingress PE transports frames received on a main interface or subinterface. This feature nullifies the need for a dummy tag and reduces overhead. In addition, frame tagging is no longer necessary.

In Ethernet port mode, both ends of a pseudowire are connected to Ethernet ports. In this mode, the port is tunneled over the pseudowire. The ingress PE transports frames received on a main interface or after the subinterface tags are removed when the packet is received on a subinterface. The VLAN manipulation is transported over the type 5 PW, whether tagged or untagged.

This figure shows a sample ethernet port mode packet flow:

Figure 2: Ethernet Port Mode Packet Flow



Configure Ethernet Port Mode

Perform this task to configure the Ethernet port mode.

```
/* PE1 configuration */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.11 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# commit

/* PE2 configuration */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/3.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.13 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the Ethernet port mode running configuration.

```
/* PE1 configuration */
l2vpn
 xconnect group grp1
  p2p xc1
   interface HundredGigE0/0/0/1.2
   neighbor 10.0.0.11 pw-id 222

/* PE2 configuration */
l2vpn
 xconnect group grp1
  p2p xc1
   interface HundredGigE0/1/0/3.2
   neighbor 10.0.0.13 pw-id 222
```

Verification

Verify the Ethernet port mode configuration.

The PW type Ethernet indicates a VC type 5 PW.

```
Router# show l2vpn xconnect group grp1 detail
Group grp1, XC xc1, state is up; Interworking none
  AC: HundredGigE0/0/0/1.2, state is up
    Type VLAN; Num Ranges: 1
    VLAN ranges: [2, 2]
    MTU 1504; XC ID 0x840006; interworking none
    Statistics:
      packets: received 186, sent 38448
      bytes: received 12644, sent 2614356
      drops: illegal VLAN 0, illegal length 0
  PW: neighbor 10.0.0.11, PW ID 222, state is up ( established )
    PW class not set, XC ID 0xc0000004
    Encapsulation MPLS, protocol LDP
    Source address 10.0.0.13
```

```
PW type Ethernet, control word disabled, interworking none
PW backup disable delay 0 sec
Sequencing not set

PW Status TLV in use
  MPLS         Local                          Remote
  ------------ ------------------------------ -----------------------------
  Label        16026                          16031
  Group ID     0x4000280                      0x6000180
  Interface    HundredGigE0/0/0/1.2           HundredGigE0/1/0/3.2
  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: 3221225476
Create time: 30/03/2021 16:30:58 (21:31:00 ago)
Last time status changed: 30/03/2021 16:36:42 (21:25:16 ago)
Statistics:
  packets: received 38448, sent 186
  bytes: received 2614356, sent 12644
```

VLAN Mode

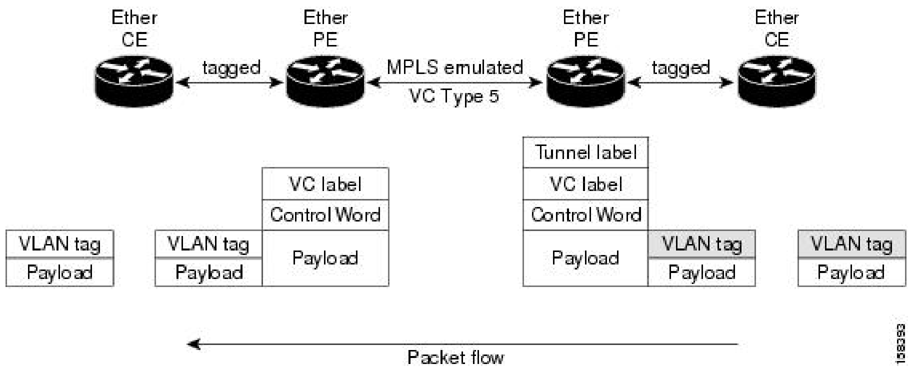
Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire VC Type 4	Release 7.3.15	With this feature, VLAN mode is supported for pseudowire over MPLS. A virtual connection (VC) type 4 is the VLAN-based PW. The ingress PE does not remove the incoming VLAN tags that are to be transported over the PW. VC type 4 inserts an extra dummy tag with VLAN 0 onto the frame which is removed on the other side. This mode helps the service provider to segregate traffic for each customer based on the VLAN.

In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4. In VLAN mode, each VLAN on a customer-end to provider-end link can be configured as a separate L2VPN connection using virtual connection (VC) type 4. VLAN-based (VC Type 4) pseudowires ensure a VLAN tag is transported over the pseudowire by pushing a dummy tag at the attachment circuit ingress. If the rewrite rule pushes two or more tags, a dummy tag is not needed because these VLAN tags are transported over the pseudowire. On the remote router, the dummy tag, if added, is removed before egress.

As illustrated in the following figure, the Ethernet PE associates an internal VLAN tag to the Ethernet port for switching the traffic internally from the ingress port to the pseudowire; however, before moving traffic into the pseudowire, it removes the internal VLAN tag.

Figure 3: VLAN Mode Packet Flow



At the egress VLAN PE, the PE associates a VLAN tag to the frames coming out of the pseudowire, and after switching the traffic internally, it sends out the traffic on an Ethernet trunk port.

Note Because the port is in trunk mode, the VLAN PE doesn't remove the VLAN tag and forwards the frames through the port with the added tag.

Limitation

On PW imposition PE, the pushed dummy VLAN Tag Tag Protocol Identifier (TPID) is copied from the TPID of the innermost VLAN tag popped on the ingress L2 interface where traffic is received from. If there is no VLAN tag popped on the L2 interface, the TPID on the dummy VLAN is 0x8100.

On the disposition PE, if the egress VLAN tag push is configured on the egress L2 interface, the innermost pushed VLAN tag TPID is copied from the TPID of the dummy VLAN tag. If there is no egress VLAN push configured on the egress L2 interface, the dummy VLAN tag is discarded.

Configure VLAN Mode

Perform this task to configure VLAN mode.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class VLAN
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode vlan
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.11 pw-id 222
Router(config-l2vpn-xc-p2p-pw)# pw-class VLAN
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the VLAN mode running configuration.

```
l2vpn
```

```
 pw-class VLAN
  encapsulation mpls
   transport-mode vlan
  !
 !
 xconnect group grp1
  p2p xc1
   neighbor 10.0.0.11 pw-id 222
    pw-class VLAN
   !
  !
 !
!
```

Verification

Verify the VLAN mode configuration.

The PW type Ethernet VLAN indicates a type 4 PW.

```
Router# show l2vpn xconnect group grp1 detail | i " PW type"
PW type Ethernet VLAN, control word disabled, interworking none
      PW type      Ethernet VLAN                  Ethernet VLAN
```

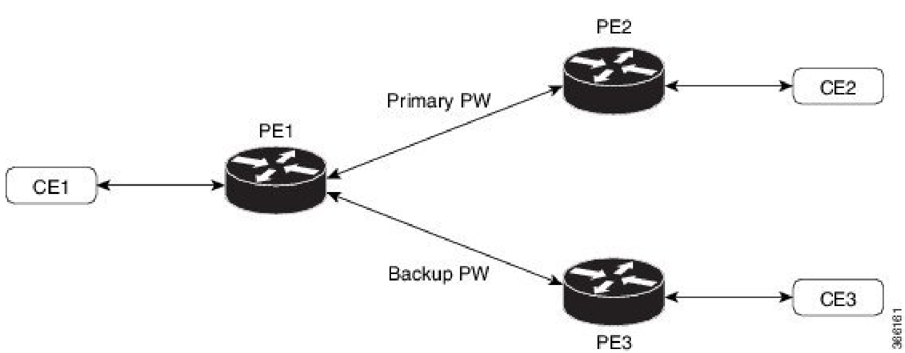
VLAN Passthrough Mode

Configure the **transport mode vlan passthrough** command under the pw-class to negotiate a virtual connection (VC)-type 4 (Ethernet VLAN) PW, which transports whatever comes out of the AC after the VLAN tag manipulation specified by the **rewrite** command.The VLAN tag manipulation on the EFP ensures that there is at least one VLAN tag left on the frame because you need a VLAN tag on the frame if there are VC-type 4 PWs. No dummy tag 0 is added to the frame when you use the **transport mode vlan passthrough** command.

Pseudowire Redundancy

The Pseudowire Redundancy feature allows you to configure a redundant pseudowire that backs up the primary pseudowire. When the primary pseudowire fails, the PE router switches to the redundant pseudowire. You can elect to have the primary pseudowire resume operation after it becomes functional. The primary pseudowire fails when the PE router fails or when there is a network outage.

Figure 4: Pseudowire Redundancy



Forcing a Manual Switchover to the Backup Pseudowire

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

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

Configure Pseudowire Redundancy

This section describes how you can configure pseudowire redundancy.

You must consider the following restrictions while configuring the Pseudowire Redundancy feature:

- 2000 active and 2000 backup PWs are supported.
- Only MPLS LDP is supported.

```
/* Configure PW on PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 172.16.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# backup neighbor 192.168.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw-backup)# commit

/* Configure PW on PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit

/* Configure PW on PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XCON1
Router(config-l2vpn-xc)# p2p xc1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/1/0/0.1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.0.0.1 pw-id 1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* On PE1 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface HundredGigE 0/1/0/0.1
   neighbor ipv4 172.16.0.1 pw-id 1
    backup neighbor 192.168.0.1 pw-id 1
!

/* On PE2 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
```

```
   interface HundredGigE 0/1/0/0.1
   neighbor ipv4 10.0.0.1 pw-id 1

!

/* On PE3 */
!
l2vpn
 xconnect group XCON1
  p2p XCON1_P2P2
   interface HundredGigE 0/1/0/0.1
   neighbor ipv4 10.0.0.1 pw-id 1

!
```

Verification

Verify that the configured pseudowire redundancy is up.

```
/* On PE1 */

Router#show l2vpn xconnect group XCON_1
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     XCON1_P2P2 UP   Hu0/1/0/0.1            UP       172.16.0.1     1000   UP
                                                           Backup
                                                           192.168.0.1     1000   SB
----------------------------------------------------------------------------------

/* On PE2 */

Router#show l2vpn xconnect group XCON_1
Tue Jan 17 15:36:12.327 UTC
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     XCON1_P2P2 UP   BE100.1                UP       10.0.0.1     1000   UP
----------------------------------------------------------------------------------------

/* On PE3 */

Router#show l2vpn xconnect group XCON_1
Tue Jan 17 15:38:04.785 UTC
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     XCON1_P2P2 DN   BE100.1                UP       10.0.0.1     1000   SB
----------------------------------------------------------------------------------------

Router#show l2vpn xconnect summary
Number of groups: 3950
Number of xconnects: 3950
```

```
  Up: 3950  Down: 0  Unresolved: 0 Partially-programmed: 0
  AC-PW: 3950  AC-AC: 0  PW-PW: 0 Monitor-Session-PW: 0
Number of Admin Down segments: 0
Number of MP2MP xconnects: 0
  Up 0 Down 0
  Advertised: 0 Non-Advertised: 0
Number of CE Connections: 0
  Advertised: 0 Non-Advertised: 0
Backup PW:
  Configured   : 3950
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 3950
  Standby Ready: 0
Backup Interface:
  Configured   : 0
  UP           : 0
  Down         : 0
  Admin Down   : 0
  Unresolved   : 0
  Standby      : 0
```

Inter-AS Mode

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Inter-AS Mode for L2VPN Pseudowire	Release 7.3.15	Inter-AS is a peer-to-peer type that allows VPNs to operate through multiple providers or multi-domain networks using L2VPN cross-connect. This mode allows VPLS autodiscovery to operate across multiple BGP autonomous systems and enables service providers to offer end-to-end VPN connectivity over different geographical locations.

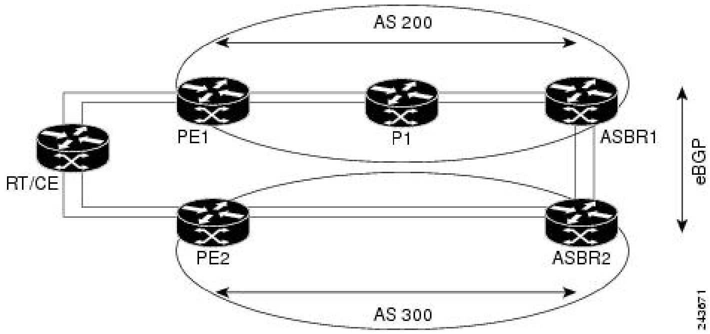
An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and uses a single, clearly defined routing protocol.

As VPNs grow, their requirements expand. In some cases, VPNs need to reside on different autonomous systems in different geographic areas. In addition, some VPNs need to extend across multiple service providers (overlapping VPNs). Regardless of the complexity and location of the VPNs, the connection between autonomous systems must be seamless.

EoMPLS supports a single AS topology where the pseudowire connecting the PE routers at the two ends of the point-to-point EoMPLS cross-connects resides in the same autonomous system; or multiple AS topologies in which PE routers can reside on two different ASs using iBGP and eBGP peering.

The following figure illustrates MPLS over Inter-AS with a basic double AS topology with iBGP/LDP in each AS.

Figure 5: EoMPLS over Inter-AS: Basic Double AS Topology



Configure Inter-AS Mode

Perform this task to configure Inter-AS mode:

```
/* PE1 Configuration */
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# router-id 10.0.0.1
Router(config-ldp)# interface HundredGigE0/2/0/3
Router(config-ldp-if)# exit
Router(config-ldp)# router bgp 100
Router(config-bgp)# bgp router-id 10.0.0.1
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group gr1
Router(config-l2vpn-xc)# mp2mp mp1
Router(config-l2vpn-xc-mp2mp)# vpn-id 100
Router(config-l2vpn-xc-mp2mp)# l2-encapsulation vlan
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# rd auto
Router(config-l2vpn-xc-mp2mp-ad)# route-target 2.2.2.2:100
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/1.1 remote-ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/1.1 remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit

/* PE2 Configuration */
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# router-id 172.16.0.1
Router(config-ldp)# interface HundredGigE0/3/0/0
Router(config-ldp-if)# exit
Router(config-ldp)# router bgp 100
Router(config-bgp)# bgp router-id 172.16.0.1
Router(config-bgp)# address-family l2vpn vpls-vpws
Router(config-bgp-af)# neighbor 10.0.0.1
Router(config-bgp-nbr)# remote-as 100
```

```
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn vpls-vpws
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group gr1
Router(config-l2vpn-xc)# mp2mp mp1
Router(config-l2vpn-xc-mp2mp)# vpn-id 100
Router(config-l2vpn-xc-mp2mp)# l2-encapsulation vlan
Router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
Router(config-l2vpn-xc-mp2mp-ad)# rd auto
Router(config-l2vpn-xc-mp2mp-ad)# route-target 2.2.2.2:100
Router(config-l2vpn-xc-mp2mp-ad)# signaling-protocol bgp
Router(config-l2vpn-xc-mp2mp-ad-sig)# ce-id 2
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/2.1 remote-ce-id 3
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# interface HunGigE0/1/0/2.2 remote-ce-id 1
Router(config-l2vpn-xc-mp2mp-ad-sig-ce)# commit
```

Running Configuration

This section shows the Inter-AS running configuration.

```
/* PE1 Configuration */
mpls ldp
 router-id 10.0.0.1
 interface HundredGigE0/2/0/3
 !
router bgp 100
 bgp router-id 10.0.0.1
 address-family l2vpn vpls-vpws
 neighbor 172.16.0.1
  remote-as 200
  update-source Loopback0
  address-family l2vpn vpls-vpws
!
l2vpn
 xconnect group gr1
  mp2mp mp1
    vpn-id 100
    l2-encapsulation vlan
    autodiscovery bgp
     rd auto
     route-target 2.2.2.2:100
     signaling-protocol bgp
      ce-id 1
       interface HunGigE0/1/0/1.1 remote-ce-id 2
       interface HunGigE0/1/0/1.2 remote-ce-id 3


/* PE2 Configuration */
mpls ldp
 router-id 172.16.0.1
 interface HundredGigE0/3/0/0
 !
router bgp 100
 bgp router-id 172.16.0.1
 address-family l2vpn vpls-vpws
 neighbor 10.0.0.1
  remote-as 100
  update-source Loopback0
  address-family l2vpn vpls-vpws
!
```

```
l2vpn
 xconnect group gr1
  mp2mp mp1
    vpn-id 100
    l2-encapsulation vlan
    autodiscovery bgp
     rd auto
     route-target 2.2.2.2:100
     signaling-protocol bgp
      ce-id 2
       interface HunGigE0/1/0/2.1 remote-ce-id 3
       interface HunGigE0/1/0/2.2 remote-ce-id 1
```

Pseudowire Headend

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
Pseudowire Headend	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) Pseudowire Headend (PWHE) is a virtual interface that allows termination of access PWs into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWHE enables integration of legacy Layer 2 services into packet-switched networks (PSNs) like IP or MPLS networks, so that users can integrate their older devices into newer networks without upgrading their hardware. This is possible because PWHE allows the termination or encapsulation of the frames from the attachment circuit into packets that can be transmitted over the PSN. * This feature is supported only on: • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Pseudowires (PWs) enable payloads to be transparently carried across IP/MPLS packet-switched networks (PSNs). PWs are regarded as simple and manageable lightweight tunnels for returning customer traffic into core networks.

Pseudowire Headend (PWHE) allows termination of access PWs into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWHE enables integration of legacy Layer 2 services into PSNs, so that users can integrate their older devices into newer networks without upgrading their hardware. This is possible because PWHE allows the termination of the frames from the attachment circuit into packets that can be transmitted over the PSN.

PWHE allows you to provision features such as QOS, access control lists (ACL), Layer 3 VPN on a per PWHE interface basis, on a Service Provider Edge (S-PE). In a typical Layer 2 VPN deployment, an attachment circuit (AC) is required to terminate the PWs. When PWHE is configured in a router, the PWHE interface emulates the behavior of an AC and terminates the PWs.

Note Only line cards and routers with the P100-based Silicon One ASIC support this feature.

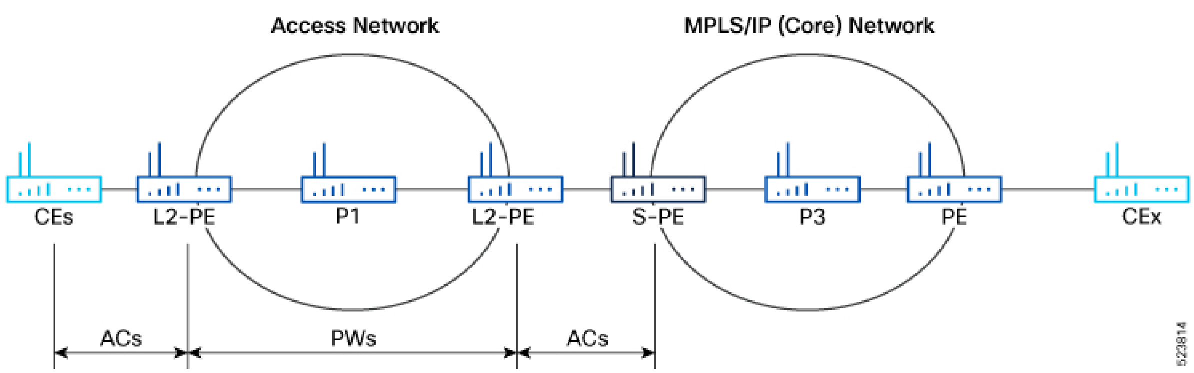
Benefits of PWHE

- Dissociates the customer facing interface (CFI) of the service PE from the underlying physical transport media of the access or aggregation network.
- Reduces capex in the access or aggregation network and service PE.
- Distributes and scales the customer facing Layer 2 user-network interfaces (UNI) set.
- Implements a uniform method of OAM functionality.
- Providers can extend or expand the Layer 3 service footprints.
- Provides a method of terminating customer traffic into a next generation network (NGN).

Topology

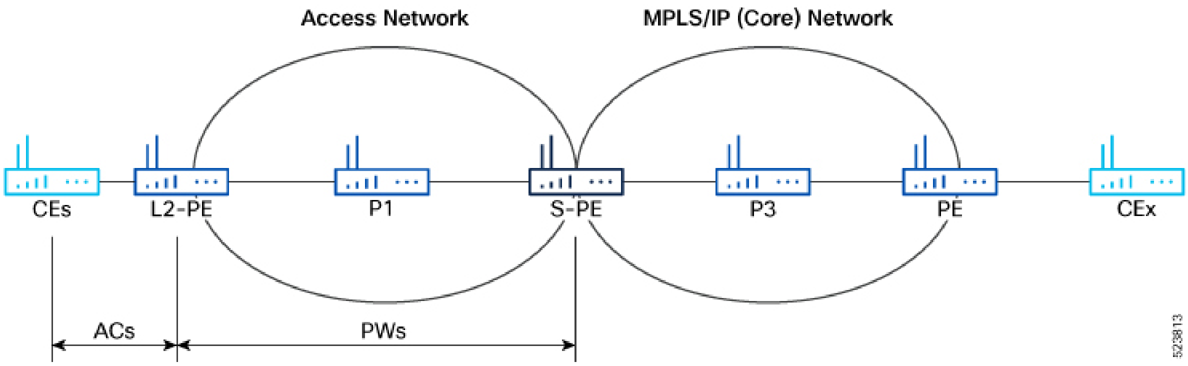
Consider a topology with Pseudowire network.

Figure 6: Pseudowire Network without PWHE



In this topology, the Layer 2 PE (L2-PE) from the access network is connected to the service provider edge (S-PE) through an AC. The PWs originating from L2-PE are terminated on the S-PE by the AC. This is a typical Layer 2 VPN deployment.

Figure 7: Pseudowire Network with PWHE



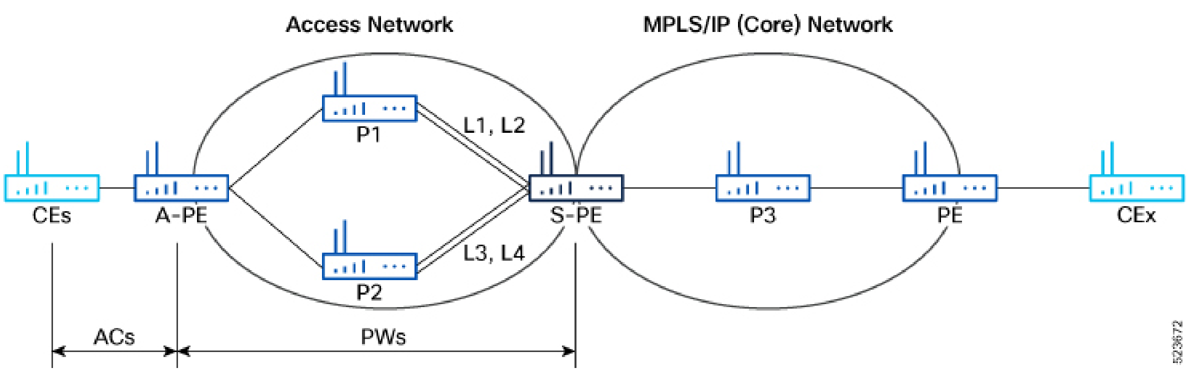
When you implement PWHE, the functionalities of L2-PE and S-PE are combined in S-PE. PWHE emulates the behavior of AC to terminate the PWs from the access network.

For PWHE cross-connect configuration, the interconnectivity between L2-PE and S-PE happens through BGP with RFC 3107 extension. The RFC 3107 allows BGP to distribute the MPLS labels along with IP prefixes. The customer network can avoid using an IGP to provide connectivity to the S-PE device, which is outside the customer's autonomous system.

PWs operate in bridged interworking mode with Virtual Circuit (VC) type 5. The VC type indicates the mode in which the packets are processed. VC type 5 is used for Ethernet port mode, where the PWs carry customer Ethernet frames (tagged or untagged) with IP payload. Thus, an S-PE device must perform ARP resolution for customer IP addresses learned over the PWHE.

Consider a topology with PWHE deployment where the access network consists of multiple provider routers.

Figure 8: PWHE Deployment



In this topology, there are multiple CEs connected to access PE (A-PE) with each CE connected by one link.

- There are two provider routers, P1 and P2, available between A-PE and S-PE in the access network.
- The S-PE is connected to P1 using links L1 and L2. The links L1 and L2 are connected to two different line cards on P1 and S-PE.
- The S-PE is connected to P2 using links L3 and L4. The links L3 and L4 are connected to two different line cards on P2 and S-PE.
- For each CE-A-PE link, a cross-connect (AC-PW) is configured on the A-PE.

While configuring mulitple PWs, group the PWs under a generic interface list (GIL). When you modify the configuration of a GIL, the changes are applied to all the PW interfaces associated with the GIL. For more information on GIL, see Generic Interface List, on page 18.

Traffic Flow Types on PWHE Interfaces

There are two types of traffic flow involved in the PWHE interfaces.

- **PWHE Decapsulation/Disposition Flow**: On traffic flowing from access side to core, PWHE ingress features would be executed on the PWHE Access facing Line card.
- **PWHE Encapsulation/Imposition Flow**: On traffic flowing from core to access side, PWHE egress features would be executed on the PWHE Access facing Line card.

PWHE Decapsulation

The packets from access side to core side are decapsulated and the ingress features are executed.

1. The outer L2 header or VLAN tag is removed from the packet that arrives at PWHE router.
2. The PW label is removed from the packet.
3. The inner L2 is removed from the packet.
4. The packet is reconstructed with the transport label, service label, and L2 header.
5. The reconstructed packet is delivered to the core network.

PWHE Encapsulation

The packets from core side to access side are encapsulated and the egress features are executed.

- The inner L2 header is encapsulated.
- The packet is reconstructed with the PW label, transport labels, and outer L2 header.
- The reconstructed packet is delivered to the access network.

Generic Interface List

A generic interface list (GIL) is a list of physical or bundle interfaces used in a PWHE connection.

The GIL supports only main interfaces, and not subinterfaces. The GIL is bi-directional and restricts both receive and transmit interfaces on access-facing line cards. The GIL has no impact on the core-facing side.

A GIL is used to limit the resources allocated for a PWHE interface to the set of interfaces specified in the list.

Only the S-PE is aware of the GIL and expects that the PWHE packets arrive on only line cards with GIL members on it. If packets arrive at the line card without GIL members on it, they are dropped.

Restrictions for Pseudowire Headend

- The following are not supported on PWHE

- ISIS as IGP for access core
- PW classVC label 4
- Segment Routing Traffic Engineering (SR-TE) in the access core for Layer 2 VPN
- TE tunnel as preferred path in access core for Layer 2 VPN
- Traffic with Internet Mix (IMIX)
- The commands **load-balancing flow src-dst-ip** and **flow-label both**
- PWHE load balancing by VC label or by Flow-Aware Transport (FAT)
- EVPN mulithoming mode
- PWHE MTU

- The load balancing hashing is done only by PWHE link numbers.
- When a packet arrives at PWHE Layer 3 subinterface, the software aggregate count is done on PWHE main interface.
- It is recommended not to use mixed-mode Egress Traffic Management (ETM), a combination of ETM and non-ETM members in a GIL.
- It is recommended not to use the ECMP path list as a superset of GIL interfaces.
- If you have configured other features like QoS and ACL on GIL, they are applicable as follows:
  - For PWHE traffic received on GIL, all the features configured on the PWHE interface are applicable.
  - For other traffic received on GIL, all the features configured on the GIL interface are applicable.
- You can attach ACL in PWHE interface for both ingress and egress, for IPv4 and IPv6.
- You can attach hybrid-ACL (Level 2) in PWHE interface for only ingress, for IPv4 and IPv6.

Configure Pseudowire Headend

Prerequisites

Consider the following guidelines while configuring PWHE:

- The generic interface list members must be the superset of the ECMP path list to the Access Provider Edge (A-PE).
- Only eight generic interface lists are supported per A-PE neighbor address.
- Eight Layer 3 links per generic interface list are supported.
- Only PW-Ether interfaces can be configured as PWHE L2 or L3 subinterfaces.
- Cross-connects that contain PW-Ether main interfaces can be configured as VC-type 5.
- PW-Ether interfaces and subinterfaces can be configured with both IPv4 and IPv6. To encapsulate and transport IPv4 or IPv6 frames over a pseudowire, the packet-switched network must be capable of routing IPv4 and IPv6 packets.

- Pseudowire redundancy, preferred path, local switching or L2TP are not supported for cross-connects configured with PWHE.
- The TE and LDP applications work on physical interfaces and therefore do not allow PWHE configuration.
- Address family, CDP, and MPLS configurations are not allowed on PWHE interfaces.
- For PWHE, eBGP, static routes, OSPF, and ISIS are supported with both IPv4 and IPv6. Routing Information Protocol (RIP) is only supported with IPv4 and not with IPv6.
- You must attach a different generic interface list for PW-Ether interfaces with different remote neighbors. Hence, you must create a separate and dedicated generic interface list for each remote neighbor or peer whose remote neighbors are different routers or devices. For example, a unique generic interface list needs to be configured for each Access Provider Edge that the Access Provider Edge peers with. The generic interface list may have the same set of outgoing interfaces.

Configuration Example

A-PE Configuration

Configure the A-PE with PWHE cross-connect to include the L2 interface and PW towards S-PE. Cross-connect or xconnect is used to establish connection between the Access Provider and Service Provider Edges.

```
/* Configure L2 interface */
Router(config)# interface hundredGigE 0/1/0/3
Router(config-if)# l2transport
Router(config-if-l2)# root

/* Configure PWHE cross-connect */

Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface hundredGigE 0/1/0/3
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 10.1.1.1 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

S-PE Configuration

Configure the S-PE:

- Create generic interface list (GIL).
- Configure pw-ether interface and attach the GIL to the interface.
- Configure cross-connect to include pw-ether interface. Use the **transport-mode ethernet** command to transport the PW traffic through Ethernet.

```
/* Create GIL */
Router(config)# generic-interface-list txlist
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

```
/* Configure pw-ether interface and attach GIL */
Router(config)# interface pw-ether1
Router(config-if)# ipv4 address 10.1.1.1/24
Router(config-if)# ipv6 address 5000::2/64
Router(config-if)# attach generic-interface-list txlist
Router(config-if)# commit

/* Configure cross-connect to include pw-ether interface */
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface pw-ether1
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.2.2.2 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

L2 Subinterface Configuration

The following example shows how to configure PWHE on L2 subinterface.

A-PE Configuration

Configure L2 subinterface.

```
/* Configure L2 subinterface */
Router(config)# interface hundredGigE 0/1/0/3.2 l2transport
Router(config-subif)# encapsulation dot1q 2
Router(config-subif)# rewrite ingress tag pop 1 symmetric
```

Note There is no need to configure cross-connect for the subinterface, as the main PWHE interface configuration is applied to the subinterface.

S-PE Configuration

Configure the L2 subinterface and add the subinterface.

```
/* Configure L2 subinterface */
Router(config)# interface PW-Ether1.2 l2transport
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# rewrite ingress tag pop 1 symmetric

/* Configure cross-connect and assign the L2 subinterface  */
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pwhe_port_vc5
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode vlan
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group pw-he
Router(config-l2vpn-xc)# p2p pw-ether1
Router(config-l2vpn-xc-p2p)# interface pw-ether1.2
Router(config-l2vpn-xc-p2p)# neighbor ipv4 10.2.2.2 pw-id 6
Router(config-l2vpn-xc-p2p-pw)# pw-class pwhe_port_vc5
Router(config-l2vpn-xc-p2p-pw)# commit
```

L3 Subinterface Configuration

The following example shows how to configure PWHE on L3 subinterface. Except the A-PE, repeat the other configurations as described for L2 interface.

A-PE Configuration

```
/* Configure L3 subinterface */
Router(config)# interface pw-ether 1.1
Router(config-subif)# encapsulation dot1q 1
Router(config-subif)# ipv4 address 10.1.1.1/24
Router(config-subif)# commit
```

Note There is no need to configure cross-connect for the subinterface, as the main PWHE interface configuration is applied to the subinterface.

Running Configuration

```
/* A-PE Configuration */
interface hundredGigE 0/1/0/3
  l2transport
!
l2vpn
   pw-class pwhe_port_vc5
   encapsulation mpls
  xconnect group pw-he
p2p pw-ether1
interface hundredGigE 0/1/0/3
neighbor ipv4 10.1.1.1 pw-id 6
pw-class pwhe_port_vc5

/* S-PE Configuration */
generic-interface-list txlist
  interface hundredGigE 0/1/0/1
  interface hundredGigE 0/1/0/2
!
interface PW-Ether1
  ipv4 address 10.1.1.1/24
  ipv6 address 5000::2/64
  attach generic-interface-list txlist
!
l2vpn
  pw-class pwhe_port_vc5
     encapsulation mpls
      transport-mode ethernet
  xconnect group pw-he
     p2p pw-ether1
       interface PW-Ether1
       neighbor ipv4 10.2.2.2 pw-id 6
         pw-class pwhe_port_vc5

/* A-PE Configuration for L2 Subinterface */
interface hundredGigE 0/1/0/3.2 l2transport
   encapsulation dot1q 2
   rewrite ingress tag pop 1 symmetric

/* S-PE Configuration for L2 Subinterface */
l2vpn
    pw-class pwhe_port_vc5
        encapsulation mpls
        transport-mode vlan
```

```
    !
    xconnect group pw-he
        p2p pw-ether1
            interface pw-ether1.2
            neighbor ipv4 10.2.2.2 pw-id 6
            pw-class pwhe_port_vc5


/* A-PE Configuration for L3 subinterface */
interface pw-ether 1.1
    encapsulation dot1q 1
    ipv4 address 10.1.1.1/24
```

Verification

The following output shows the details of GIL.

```
Router# show generic-interface-list idb name txlist

GIL name: txlist, ifhandle: 0xf000014
  State: Down Immediate
  Members:
    HundredGigE0/1/0/2
    HundredGigE0/1/0/1
```

The following output shows the status of xconnect group with PWHE Up.

```
Router# show l2vpn xconnect group pw-he xc-name pw-ether1
Mon Mar 11 21:26:48.281 EDT
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
SB = Standby, SR = Standby Ready, (PP) = Partially Programmed,
LU = Local Up, RU = Remote Up, CO = Connected, (SI) = Seamless Inactive

XConnect                  Segment 1       Segment 2
Group     Name        ST  Description ST Description              ST
------------------------ ----------------------------- -----------------------------
pw-he pw-ether1       UP PE1          UP EVPN 1,1,102.102.102.102 UP
----------------------------------------------------------------------------------------
```

The following output shows the status of PWHE interface with PWHE Up.

```
Router# show l2vpn pwhe interface pw-ether 1 detail

Interface: PW-Ether1 Interface State: Up, Admin state: Up
Interface handle 0xf000054
MTU: 1514
BW: 10000 Kbit
Interface MAC addresses: 1859.f57d.0008
Label: 24041
Internal ID: None
L2-overhead: 0
VC-type: 5
CW: Y
Hash: 0xf5f5 [Success]
```

Traffic Mirroring on PWHE

Table 6: Feature History Table

Feature Name	Release Information	Feature Description

Traffic Mirroring on PWHE	Release 24.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: P100]) (select variants only*) This feature allows you to perform a detailed inspection and analysis of layer 2 network traffic passing through a set of ethernet interfaces without interrupting the flow of traffic. You can copy or mirror the network packets that pass through a specific source interface within a layer 2 VPN, allowing these packets to be redirected to a predetermined destination interface. *This feature is supported on: • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E

Traffic mirroring, also known as Switched Port Analyzer (SPAN) is a traffic monitoring and analysis tool that enables a user to monitor the network traffic passing through a set of ethernet interfaces.

Cisco IOS XR Release 24.3.1 introduces traffic mirroring on PWHE. This allows the mirroring of packets that pass through a source interface to a specified destination interface. The destination interface may then be attached to a network analyzer for debugging. For more information about the Traffic Mirroring, see the *Configuring Traffic Mirroring* chapter in the *Interface and Hardware Component Configuration Guide for Cisco 8000 Series Routers*. For complete command reference for this feature, see the *Traffic Mirroring Commands* chapter in the *Interface and Hardware Component Command Reference for Cisco 8000 Series Routers*.

SPAN feature support on PWHE

PWHE supports these SPAN features:

- ERSPAN: This feature allows you to monitor traffic over an IP network to a remote monitoring device. You can analyze the traffic from a switch that is not locally connected to the monitoring device.
- Security ACL with or without UDF: This feature allows you to use the Access Control Lists (ACLs) to filter network traffic based on certain criteria. UDFs can be used within ACLs to create more granular and customized filtering rules.
- Forward drop mirroring: This feature allows you to copy or mirror packets that are dropped during the forwarding process at the router ingress to a configured destination. These mirrored packets can be captured and analyzed using network monitoring tools. The analysis of dropped packets helps you understand the types of traffic that are blocked, analyze potential security threats, troubleshoot, and optimize network performance.
- Truncation: This feature allows you to capture only a portion of each packet using a monitoring tool. This is useful for reducing the amount of data that needs to be processed and stored during network analysis.
- SPAN to file: This feature allows you to captured traffic to be written directly to a file for later analysis. This is useful for archiving data or for situations where real-time analysis is not required.
- Buffer drop mirroring: This feature allows you to mirrors packets that are dropped at the ingress buffer of a switch. This is useful for diagnosing network congestion problems.

For configuring traffic mirroring on PWHE, see the Configuring Traffic Mirroring Features on the PWHE Interface, on page 25.

Limitations and Restrictions for Traffic Mirroring on PWHE

These are the limitations and restrictions of traffic mirroring on PWHE:

- PWHE don't support these SPAN features:
  - Egress SPAN
  - Local SPAN
  - Span ACL
  - Mixed mode SPAN
  - Multi SPAN ACL
- The truncation supports capturing packet sizes up to 176 bytes for IPv4 traffic and 196 bytes for IPv6 traffic.
- Linecards and fixed routers with Q200 and P100 based Silicon One ASICs support buffer-drop mirroring.

Configuring Traffic Mirroring Features on the PWHE Interface

You can configure any of these SPAN features on the PWHE interface as per your specific network monitoring and troubleshooting requirements:

- ERSPAN. Refer to Configuring ERSPAN on the PWHE Interface, on page 25.
- Security ACL. Refer to Configuring Security ACL on the PWHE Interface, on page 28.
- Forward drop mirroring. Refer to Configuring Forward Drop Mirroring on the PWHE Interface, on page 30.
- Truncation. Refer to Configuring Truncation on the PWHE Interface, on page 32.
- SPAN to file. Refer to Configuring Span to File on the PWHE Interface, on page 33.
- Buffer drop mirroring. Refer to Configuring Buffer Drop Mirroring on the PWHE Interface, on page 34.

Configuring ERSPAN on the PWHE Interface

Perform these steps to configure the ERSPAN on the PWHE interface:

Step 1 Configure a Generic Interface List (GIL) with the **generic-interface-list** command to include multiple interfaces within a generic interface.

Example:

```
Router# configure
Router(config)# generic-interface-list gi1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

Step 2 Configure a pseudowire interface with the **interface pw-ether** command and attach a GIL to the interface to encapsulate and transport the mirrored traffic.

Example:

```
Router# configure
Router(config)# interface pw-ether1
Router(config-if)# mac-address aa39.3362.3007
Router(config-if)# ipv4 address 200.0.0.1 255.255.255.0
Router(config-if)# attach generic-interface-list gi1
```

Step 3 Configure a traffic monitoring session with the **monitor-session** command to monitor the inbound traffic.

Example:

```
Router(config-if)# monitor-session mysession ethernet direction rx-only
Router(config-if)# commit
```

Step 4 View the running configuration to verify the configuration that you have configured.

Example:

```
/* Create GIL */
generic-interface-list gi1
    interface hundredGigE 0/1/0/1
    interface hundredGigE 0/1/0/2
!
/* Configure pw-ether interface and attach GIL */
interface PW-Ether1
    mac-address aa39.3362.3007
    ipv4 address 200.0.0.1 255.255.255.0
    attach generic-interface-list gi1

/* Apply traffic monitoring session to the pw-ether interface*/
    monitor-session mysession ethernet direction rx-only
!
```

Step 5 Verify the ERSPAN configuration on the PWHE interface with the **show monitor-session <session_name> status** command and **show monitor-session <session_name> status internal** command for session statistics.

In the following example, you can verify the ERSPAN configuration on the PWHE interface with the **show monitor-session <session_name> status** command.

```
Router# show monitor-session status
Monitor-session mon1
Destination interface tunnel-ip1
================================================================================
Source Interface            Dir         Status
---------------------      ----  ----------------------------------------------------
Hu0/1/0/14                  Rx    Operational
Hu0/1/0/15.100              Rx    Operational
BE1                         Rx    Operational
BE1.1                       Rx    Operational
```

In the following example, you can verify the ERSPAN configuration on the PWHE interface with the **show monitor-session <session_name> status internal** command.

```
Router# show monitor-session status internal
Information from SPAN Manager and MA on all nodes:
Monitor-session mon1 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface <>(0x00800190)
Last error: Success
0/1/CPU0: Destination interface <>(0x00800190)
0/RP0/CPU0: Destination interface <>(0x00800190)
```

```
Information from SPAN EA on all nodes:
Monitor-session 0x00000001 (Ethernet)
0/1/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/1/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 32
Monitor Session Bytes: 4024
0/2/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/2/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 0
Monitor Session Bytes: 0
```

Step 6 Capture the packets using a traffic generator tool.

Step 7 Verify that the captured packet is a GRE packet and should match the original sent packet encapsulated with a GRE header.

Example:

The following example of a decoded ERSPAN captured packet displays that the original packet is encapsulated with a GRE header:

```
/*Captured GRE packet*/
Frame 13: 1330 bytes on wire (10640 bits), 1330 bytes captured (10640 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 30, 2023 16:06:03.167332961 EST
    UTC Arrival Time: Nov 30, 2023 21:06:03.167332961 UTC
    Epoch Arrival Time: 1701378363.167332961
    [Time shift for this packet: 0.000000000 seconds]
    [Time delta from previous captured frame: 0.000001910 seconds]
    [Time delta from previous displayed frame: 0.000001910 seconds]
    [Time since reference or first frame: 0.500001960 seconds]
    Frame Number: 13
    Frame Length: 1330 bytes (10640 bits)
    Capture Length: 1330 bytes (10640 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:gre:erspan:eth:ethertype:ip:data]

/*GRE header*/
Ethernet II, Src: Cisco_75:50:dc (34:88:18:75:50:dc), Dst: Cisco_00:00:01 (00:12:01:00:00:01)
    Destination: Cisco_00:00:01 (00:12:01:00:00:01)
    Source: Cisco_75:50:dc (34:88:18:75:50:dc)
    Type: IPv4 (0x0800)
    Frame check sequence: 0x41b15e2b [unverified]
    [FCS Status: Unverified]
Internet Protocol Version 4, Src: 7.0.0.1, Dst: 7.0.0.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1312
    Identification: 0x0000 (0)
    010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 254
    Protocol: Generic Routing Encapsulation (47)
    Header Checksum: 0x69ac [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 7.0.0.1
    Destination Address: 7.0.0.2
Generic Routing Encapsulation (ERSPAN)
Encapsulated Remote Switch Packet ANalysis Type II
```

```
/*Original packet*/
Ethernet II, Src: Cisco_00:ee:00 (28:af:fd:00:ee:00), Dst: Cisco_75:50:ec (34:88:18:75:50:ec)
    Destination: Cisco_75:50:ec (34:88:18:75:50:ec)
    Source: Cisco_00:ee:00 (28:af:fd:00:ee:00)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 13.0.0.2, Dst: 7.0.0.2
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1262
    Identification: 0x0000 (0)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 63
    Protocol: any host internal protocol (61)
    Header Checksum: 0x62d0 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 13.0.0.2
    Destination Address: 7.0.0.2
Data (1242 bytes)
    Data [truncated]: ecc0d6806f7683004978696060000000101112139c1d0b9904be1a1b1c1d1e1f202
                      122232425262728292a2b2c2d2e2f303132333435363738393a3b3c3d3e3f404142
                      434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f606162636
                      465666768696a6b6c6d6
    [Length: 1242]
```

Configuring Security ACL on the PWHE Interface

Perform these steps to configure the security ACL on the PWHE interface:

Step 1 Configure a Generic Interface List (GIL) with the **generic-interface-list** command to include multiple interfaces within a generic interface.

Example:

```
Router# configure
Router(config)# generic-interface-list gi1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/1
Router(config-gen-if-list)# interface hundredGigE 0/1/0/2
Router(config-gen-if-list)# commit
```

Step 2 Configure a pseudowire interface with the **interface pw-ether** command and attach a GIL to the interface to encapsulate and transport the mirrored traffic.

Example:

```
Router# configure
Router(config)# interface pw-ether1
Router(config-if)# mac-address aa39.3362.3007
Router(config-if)# ipv4 address 200.0.0.1 255.255.255.0
Router(config-if)# attach generic-interface-list gi1
Router(config-if)# commit
```

Step 3 Configure security ACL with the **ipv4 access-list span-acl** command to create a specific ACL that can be applied to monitor and control traffic on the PWHE interface.

Example:

```
Router# configure
Router(config)# ipv4 access-list span-acl1
Router(config)# 10 permit ipv4 any host 54.0.0.2 capture
Router(config)# commit
```

Step 4 Apply the security ACL session to the configured pseudowire interface with the **monitor-session** command to monitor inbound traffic and apply an ACL to filter that traffic.

Example:

```
Router# configure
Router(config)# interface pw-ether1
Router(config-if)# monitor-session span0 ethernet direction rx-only ac
Router(config-if)# ipv4 access-group span-acl0 ingress
Router(config-if)# commit
```

Step 5 View the running configuration to verify the configuration that you have configured.

Example:

```
/* Create GIL */
generic-interface-list gi1
    interface hundredGigE 0/1/0/1
    interface hundredGigE 0/1/0/2
!
/* Configure pw-ether interface and attach GIL */
interface PW-Ether1
    mac-address aa39.3362.3007
    ipv4 address 200.0.0.1 255.255.255.0
    attach generic-interface-list gi1
!
/*Configure ACL*/
ipv4 access-list span-acl1
    10 permit ipv4 any host 54.0.0.2 capture

/* Apply ACL session to the pw-ether interface*/
    interface pw-ether1
    monitor-session span0 ethernet direction rx-only ac
    ipv4 access-group span-acl0 ingress
!
```

Step 6 Verify the security ACL configuration on the PWHE interface with the **show monitor-session <session-name> status** command and **show monitor-session <session-name> status internal** command for session statistics.

In the following example, you can verify the security ACL configuration on the PWHE interface using the **show monitor-session <session-name> status** command.

```
Router# show monitor-session status
Monitor-session mon1
Destination interface tunnel-ip1
================================================================================
Source Interface            Dir         Status
---------------------      ----  ----------------------------------------------------
Hu0/1/0/14                  Rx    Operational
Hu0/1/0/15.100              Rx    Operational
BE1                         Rx    Operational
BE1.1                       Rx    Operational
```

In the following example, you can verify the security ACL configuration on the PWHE interface with the **show monitor-session <session-name> status internal** command.

```
Router# show monitor-session status internal
Thu Aug 13 20:05:23.478 UTC
Information from SPAN Manager and MA on all nodes:
```

```
Monitor-session mon1 (ID 0x00000001) (Ethernet)
SPAN Mgr: Destination interface <>(0x00800190)
Last error: Success
0/1/CPU0: Destination interface <>(0x00800190)
0/RP0/CPU0: Destination interface <>(0x00800190)
Information from SPAN EA on all nodes:
Monitor-session 0x00000001 (Ethernet)
0/1/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/1/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 32
Monitor Session Bytes: 4024
0/2/CPU0: Name 'mon1', destination interface <> (0x00800190)
Platform, 0/2/CPU0:
Monitor Session ID: 1
Monitor Session Packets: 0
Monitor Session Bytes: 0
```

Configuring Forward Drop Mirroring on the PWHE Interface

Perform these steps to configure the forward drop mirroring on the PWHE interface:

Step 1 Configure a traffic monitoring session with the **monitor-session** command.

ERSPAN and span to file features supports the forward drop mirroring. You can configure a traffic monitoring session with the ERSPAN or span to file feature to enable forward drop mirroring.

- Configure a traffic monitoring session with span to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
Router(config-mon)# commit
```

- Configure a traffic monitoring session with ERSPAN to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination tunnel-ip1
```

Step 2 Capture packet drops with the **forward-drop** or **drops packet-processing rx** command to capture packets at all ports on the network forwarding module.

Example:

```
Router(config-mon)# forward-drop rx
```

Step 3 View the running configuration to verify the configuration that you have configured.

Example:

```
/*configure a traffic monitoring session*/
monitor-session mysession ethernet
    destination file
    forward-drop rx
!
```

Step 4 Verify the forward drop mirroring configuration on the PWHE interface with the **show spp node-counters** command.

In the following example, you can verify the froward drop mirroring configuration on the PWHE interface.

```
Router# show spp node-counters
Tue Feb 27 21:12:49.886 UTC
0/1/CPU0:
socket/rx
                ether raw pkts:              10
                low que accept:              10
           sent to XR classify:              10
-------------------------------
socket/tx
                       ce pkts:              20
        SW padding vector used:              20
-------------------------------
device/classify
      forwarded to spp clients:              10
 forwarded NPU packet to NetIO:              10
            L3 Route not found:              10
-------------------------------
client/inject
        pkts injected into spp:              10
  NetIO->CPU injected into spp:              10
  NetIO->CPU PKT IPV4_PREROUTE:              10
-------------------------------
pd_span_drop
                     SPAN drop:              10
-------------------------------
client/punt
              punted to client:              10
-------------------------------
```

Step 5 Perform these steps to capture and verify packets.

- Perform these steps to capture and verify forward drop mirroring using span to file feature:

a) Enable capturing the forward drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

b) Stop capturing the forward drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

c) Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

d) Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

e) Verify that the captured packet matches the original packet.

- Perform these steps to capture and verify forward drop mirroring using ERSPAN feature:

a) Capture the packets with a traffic generator tool.
b) Verify that the captured packet is a GRE packet and should match the original sent packet encapsulated with a GRE header.

Configuring Truncation on the PWHE Interface

Perform these steps to configure the truncation on the PWHE interface:

Step 1 Configure a traffic monitoring session with the **monitor-session** command to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
```

Step 2 Truncate the mirrored packets to a specific length with the **mirror first** command to capture only the required information for monitoring and troubleshooting.

Example:

```
Router(config-mon)# mirror first 128
 Router(config-mon)# commit
```

Step 3 Configure the traffic monitoring session with the **monitor-session** command to monitor the inbound traffic.

Example:

```
Router(config-if)# monitor-session mysession ethernet direction rx-only
```

Step 4 View the running configuration to verify the configuration that you have configured.

Example:

```
/*configure a traffic monitoring session*/
monitor-session mysession ethernet
    destination file

/*Truncate the mirrored packets to a specific length*/
    mirror first 128
!
/*Configure the traffic monitoring session to monitor the inbound traffic*/
monitor-session mysession ethernet direction rx-only
```

Step 5 Verify the truncation configuration on the PWHE interface with the **show spp node-counters** command.

In the following example, you can verify the truncation configuration on the PWHE interface.

```
Router# show spp node-counters
Tue Feb 27 21:12:49.886 UTC
0/1/CPU0:
socket/rx
                ether raw pkts:              10
                low que accept:              10
           sent to XR classify:              10
```

```
-------------------------------
socket/tx
                      ce pkts:             20
       SW padding vector used:             20
-------------------------------
device/classify
     forwarded to spp clients:             10
 forwarded NPU packet to NetIO:            10
           L3 Route not found:             10
-------------------------------
client/inject
       pkts injected into spp:             10
  NetIO->CPU injected into spp:            10
  NetIO->CPU PKT IPV4_PREROUTE:            10
-------------------------------
pd_span_drop
                    SPAN drop:             10
-------------------------------
client/punt
            punted to client:              10
-------------------------------
```

Step 6 Enable capturing the truncation packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

Step 7 Stop capturing the truncation packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

Step 8 Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

Step 9 Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

Configuring Span to File on the PWHE Interface

Perform these steps to configure the span to file on the PWHE interface:

Step 1 Configure a traffic monitoring session with the **monitor-session** command to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
Router(config-mon)# commit
```

Step 2 Configure the traffic monitoring session with the **monitor-session** command to monitor the inbound traffic.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet direction rx-only
```

Step 3 View the running configuration to verify the configuration that you have configured.

Example:

```
Router# show running-config interface hundredGigE 0/1/0/0
interface HundredGigE0/1/0/0
 ipv4 address 10.0.0.1 255.255.255.0
 monitor-session mysession ethernet direction rx-only
```

Step 4 Verify the the span to file configuration on the PWHE interface with the **show spp node-counter** command.

In the following example, you can verify the span to file configuration on the PWHE interface.

```
Router# show spp node-counters location 0/5/CPU0  | i SPAN
                  SPAN to File:          299846
```

Step 5 Enable capturing the span to file packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

Step 6 Stop capturing the span to file packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

Step 7 Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

Step 8 Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

Configuring Buffer Drop Mirroring on the PWHE Interface

Only the linecards and fixed routers with Q200 and P100 based silicon one ASICs support buffer drop mirroring.

Perform these steps to configure the buffer drop (TM drop) mirroring on the PWHE interface.

Step 1 Configure a traffic monitoring session with the **monitor-session** command.

ERSPAN and span to file features supports the buffer drop mirroring. You can configure a traffic monitoring session for buffer drop mirroring with the ERSPAN or span to file feature.

- Configure a traffic monitoring session with span to file to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination file
Router(config-mon)# commit
```

- Configure a traffic monitoring session with ERSPAN to capture and save mirrored traffic to a file on the router.

Example:

```
Router# configure
Router(config)# monitor-session mysession ethernet
Router(config-mon)# destination tunnel-ip1
```

Step 2 Capture packet drops with the **tm-drop** or **drops traffic-management rx** command to capture packets at all ports on the network forwarding module.

Example:

```
Router(config-mon)# tm-drop rx
```

Step 3 View the running configuration to verify the configuration that you have configured.

Example:

```
/*configure a traffic monitoring session*/
monitor-session mysession ethernet
    destination file
    tm-drop rx
!
```

Step 4 Verify the buffer drop mirroring configuration on the PWHE interface with the **show spp node-counters** command.

In the following example, you can verify the buffer drop mirroring configuration on the PWHE interface.

```
Router# show spp node-counters
Tue Feb 27 21:12:49.886 UTC
0/1/CPU0:
socket/rx
                ether raw pkts:              10
                low que accept:              10
           sent to XR classify:              10
-------------------------------
socket/tx
                       ce pkts:              20
        SW padding vector used:              20
-------------------------------
device/classify
      forwarded to spp clients:              10
 forwarded NPU packet to NetIO:              10
            L3 Route not found:              10
-------------------------------
client/inject
        pkts injected into spp:              10
```

```
  NetIO->CPU injected into spp:              10
  NetIO->CPU PKT IPV4_PREROUTE:              10
-------------------------------
pd_span_drop
                     SPAN drop:              10
-------------------------------
client/punt
              punted to client:              10
-------------------------------
```

Step 5 Perform these steps to capture and verify packets.

- Perform these steps to capture and verify buffer drop mirroring with the span to file feature:

a) Enable capturing the buffer drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection start
```

The router saves the pcap file in the temporary folder.

b) Stop capturing the buffer drop mirroring packets with the **monitor-session** command.

Example:

```
Router# monitor-session mysession packet-collection stop write directory myfoldername filename
myfilename
```

The router saves the pcap file in the specified folder.

Note This command doesn't create a folder; you must enter an existing folder in the router.

c) Enter shell mode with the **run** command and navigate to the folder where the pcap file is saved.

Example:

```
Router# run
[node0_RP0_CPU0:~]$ cd /myfoldername/node0_1_CPU0/
[node0_RP0_CPU0:/myfoldername/node0_1_CPU0]$ ls
myfilename.pcap
```

d) Use remote file copy commands like **scp** from your lab server to copy the pcap files from router to your local computer and view the pcap file.

e) Verify that the captured packet matches the original packet.

- Perform these steps to capture and verify buffer drop mirroring with ERSPAN feature:

a) Capture the packets with a traffic generator tool.

b) Verify that the captured packet is a GRE packet and should match the original sent packet encapsulated with a GRE header.

Preferred Tunnel Path

Table 7: Feature History Table

Feature Name	Release Information	Feature Description

VPLS over Preferred TE and MPLS OAM	Release 7.5.2	Based on your network traffic pattern, you can configure the preferred Traffic Engineering (TE) tunnel path between Provider Edge (PE) routers participating in the same Virtual Private LAN Services (VPLS). You optimize network resource utilization and performance when you set an explicit path on the PE router to direct traffic flow to a specific destination PE router. With VPLS, you now have MPLS-OAM capabilities for troubleshooting MPLS networks: • MPLS LSP Ping • MPLS LSP Traceroute • Flow-Aware Transport (FAT) Pseudowires (PW) This functionality adds the following command: control-word

Preferred tunnel path functionality lets you map pseudowires to specific traffic-engineering tunnels. Attachment circuits are cross-connected to specific MPLS traffic engineering tunnel interfaces instead of remote PE router IP addresses (reachable using IGP or LDP). Using preferred tunnel path, it is always assumed that the traffic engineering tunnel that transports the L2 traffic runs between the two PE routers (that is, its headend starts at the imposition PE router and its tailend terminates on the disposition PE router).

Note

- Currently, preferred tunnel path configuration applies only to MPLS encapsulation.

Configure Preferred Tunnel Path

Configuration Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class PATH1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path interface tunnel-te 11 fallback disable
Router(config-l2vpn-pwc-mpls)# commit
```

Tunnel Configuration for VPLS

```
interface tunnel-te1
 ipv4 unnumbered Loopback0
 signalled-bandwidth 50
 destination 10.12.12.12
 path-option 1 explicit name FC1
```

L2 Configuration Example—VPLS over Preferred TE Tunnel

To configure VPLS over preferred TE tunnel, run the following commands:

```
RP/0/RP0/CPU0:r1(config)#interface FourHundredGigE0/0/0/0.1 l2transport
RP/0/RP0/CPU0:r1(config-subif)#encapsulation dot1q 100
RP/0/RP0/CPU0:r1(config-subif)#rewrite ingress tag pop 1 symmetric
RP/0/RP0/CPU0:r1(config-subif)#exit
RP/0/RP0/CPU0:r1(config)#l2vpn
RP/0/RP0/CPU0:r1(config)#pw-class c
RP/0/RP0/CPU0:r1(config-l2vpn-pwc)#encapsulation mpls
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#control-word
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#load-balancing
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls-load-bal)#flow-label both
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls-load-bal)#exit
RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#preferred-path interface tunnel-te 1

RP/0/RP0/CPU0:r1(config-l2vpn-pwc-mpls)#exit
RP/0/RP0/CPU0:r1(config-l2vpn-pwc)#exit
RP/0/RP0/CPU0:r1(config-l2vpn)#bridge group bg bridge-domain bd
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd)#interface FourHundredGigE0/0/0/0.1
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd-ac)#exit
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd)#neighbor 10.12.12.12 pw-id 100
RP/0/RP0/CPU0:r1(config-l2vpn-bg-bd-pw)#pw-class c
```

Verification

```
RP/0/RP0/CPU0:r1#show l2vpn bridge-domain detail
Wed Apr 20 17:53:26.232 UTC
Legend: pp = Partially Programmed.
Bridge group: bg, bridge-domain: bd, id: 0, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  VINE state: Default
  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: 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: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 1
  Filter MAC addresses:
  P2MP PW: disabled
  Multicast Source: Not Set
  Create time: 20/04/2022 17:37:30 (00:15:55 ago)
  No status change since creation
  ACs: 1 (1 up), VFIs: 0, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: FourHundredGigE0/0/0/0, state is up
      Type Ethernet
```

```
      MTU 1500; XC ID 0x1; interworking none
      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: bridge-domain policer
      Static MAC addresses:
      Statistics:
        packets: received 0 (multicast 0, broadcast 0, unknown unicast 0, unicast 0), sent
0
        bytes: received 0 (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: Learn key: 0
List of Access PWs:
   PW: neighbor 10.12.12.12, PW ID 100, state is up ( established )
      PW class c, XC ID 0xa0000001
      Encapsulation MPLS, protocol LDP
      Source address 10.10.10.10
      PW type Ethernet, control word enabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
      Preferred path Active : tunnel-te1, Statically configured, fallback enabled
      Ignore MTU mismatch: Disabled
      Transmit MTU zero: Disabled
      Tunnel : Up

      PW Status TLV in use
        MPLS         Local                          Remote
        ------------ ------------------------------ ---------------------------
        Label        24000                          24000
        Group ID     0x0                            0x0
        Interface    Access PW                      Access PW
        MTU          1500                           1500
        Control word enabled                        enabled
        PW type      Ethernet                       Ethernet
        VCCV CV type 0x2                            0x2
                     (LSP ping verification)        (LSP ping verification)
        VCCV CC type 0x7                            0x7
                     (control word)                 (control word)
                     (router alert label)           (router alert label)
                     (TTL expiry)                   (TTL expiry)
        ------------ ------------------------------ ---------------------------
      Incoming Status (PW Status TLV):
```

```
  Status code: 0x0 (Up) in Notification message
MIB cpwVcIndex: 2684354561
Create time: 20/04/2022 17:37:30 (00:15:55 ago)
Last time status changed: 20/04/2022 17:53:22 (00:00:04 ago)
MAC withdraw messages: sent 0, received 0
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: 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
DHCPv4 Snooping: disabled
DHCPv4 Snooping profile: none
IGMP Snooping: disabled
IGMP Snooping profile: none
MLD Snooping profile: none
Storm Control: bridge-domain policer
```

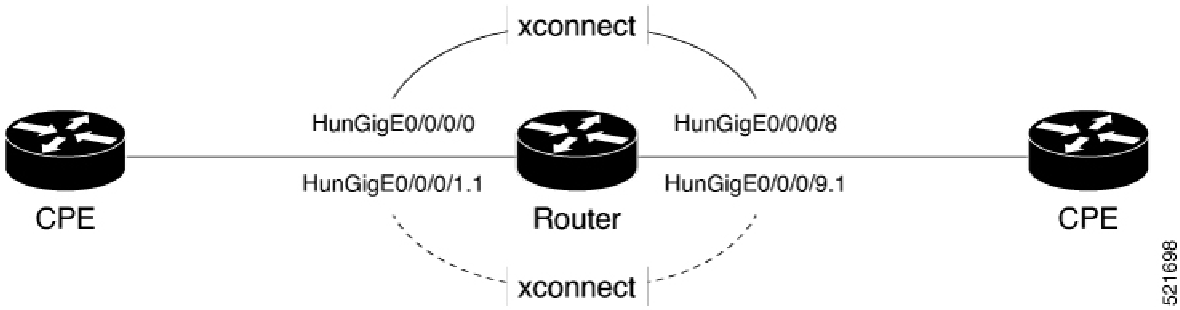
Configure Local Switching Between Attachment Circuits

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Support of Tagged or Untagged VLAN on Physical and Bundle AC with VLAN Rewrite	Release 7.3.15	This feature supports tagged or untagged VLAN on physical and bundle interfaces. The tagged VLAN allows you to send and receive the traffic for multiple VLANs whereas, the untagged VLAN allows you to send and receive the traffic for a single VLAN. The multiple VLANs are used to differentiate traffic streams so that the traffic can be split across different services.

Local switching involves the exchange of L2 data from one attachment circuit (AC) to the other. The two ports configured in a local switching connection form an attachment circuit (AC). A local switching connection works like a bridge domain that has only two bridge ports, where traffic enters from one port of the local connection and leaves through the other.

Figure 9: Local Switching Between Attachment Circuits



These are some of the characteristics of Layer 2 local switching:

- Because there is no bridging involved in a local connection, there is neither MAC learning nor flooding.
- ACs in a local connection are not in the UP state if the interface state is DOWN.
- Local switching ACs utilize a full variety of Layer 2 interfaces, including Layer 2 trunk (main) interfaces, bundle interfaces, and Ethernet Flow Points (EFPs).
- Same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

Configuration

To configure an AC-AC point-to-point cross connect, complete the following configuration:

- Create Layer 2 interfaces.
- Create a cross-connect group and point-to-point connection.
- Attach the Layer 2 interfaces to point-to-point connection.

```
/* Configure L2 transport and encapsulation on the VLAN sub-interfaces */
Router# configure
Router(config)# interface HunGigE 0/0/0/1.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# exit
Router(config)# interface HunGigE 0/0/0/9.1 l2transport
Router(config-subif)# encapsulation dot1q 5
Router(config-subif)# commit
```

```
/* Configure local switching on the VLAN sub-interfaces */
Router(config)# l2vpn
Router(config-l2vpn-xc)# p2p XCON1_P2P1
Router(config-l2vpn-xc-p2p)# interface HunGigE0/0/0/1.1
Router(config-l2vpn-xc-p2p)# interface HunGigE0/0/0/9.1
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit
```

Running Configuration

```
configure


interface HunGigE 0/0/0/1.1 l2transport
 encapsulation dot1q 5
!
interface HunGigE 0/0/0/9.1 l2transport
  encapsulation dot1q 5
 !


l2vpn
   p2p XCON1_P2P1
    interface HunGigE0/0/0/1.1
    interface HunGigE0/0/0/9.1
    !
   !
 !
```

Verification

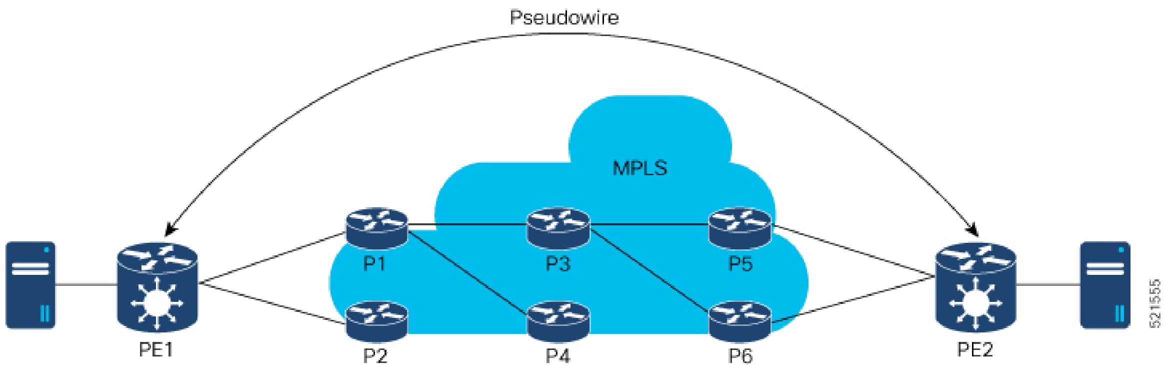
- Verify if the configured cross-connect is UP

```
router# show l2vpn xconnect brief

Locally Switching

  Like-to-Like                        UP        DOWN        UNR

    EFP                                1           0           0

    Total                              1           0           0


  Total                                1           0           0

Total: 1 UP, 0 DOWN, 0 UNRESOLVED
```

MPLS PW Traffic Load Balancing on P Router

When an L2VPN PE needs to send a frame over an MPLS PW, the Ethernet frame is encapsulated into an MPLS frame with one or more MPLS labels; there is at least one PW label and perhaps an IGP label to reach the remote PE.

The MPLS frame is transported by the MPLS network to the remote L2VPN PE. There are typically multiple paths to reach the destination PE:



PE1 can choose between P1 and P2 as the first MPLS P router towards PE2. If P1 is selected, P1 then chooses between P3 and P4, and so on. The available paths are based on the IGP topology and the MPLS TE tunnel path.

MPLS service providers prefer to have all links equally utilized rather than one congested link with other underutilized links. This goal is not always easy to achieve because some PWs carry much more traffic than others and because the path taken by a PW traffic depends upon the hashing algorithm used in the core. Multiple high bandwidth PWs might be hashed to the same links, which creates congestion.

A very important requirement is that all packets from one flow must follow the same path. Otherwise, this leads to out-of-order frames, which might impact the quality or the performance of the applications.

Use the following methods to load balance the MPLS PW traffic:

Load Balance MPLS PW Traffic using Control-Word and Flow-Label

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Load Balance MPLS PW Traffic using Control-Word	Release 7.3.15	This feature allows the router to correctly identify the Ethernet PW packet over an IP packet, thus preventing the selection of wrong equal-cost multipath (ECMP) path for the packet that leads to the misordering of packets. This feature inserts the control word keyword immediately after the MPLS label to separate the payload from the MPLS label over a PW. The control word carries layer 2 control bits and enables sequencing. The **control-word** keyword is added.

Load Balance MPLS PW Traffic using Flow-Label	Release 7.3.15	The flow-label provides the capability to identify individual flows within a pseudowire and provides routers the ability to use these flows to load balance traffic. Individual flows are determined by the hashing algorithm configured under L2VPN. Similar packets with the same source and destination addresses are all said to be in the same flow. A flow-label is created based on indivisible packet flows entering a pseudowire and is inserted as the lowermost label in the packet. Routers can use the flow-label for load balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core. The **flow-label** keyword is added.

Load balancing using Control-Word

If the MPLS packet contains the MAC address that starts with 0x4 or 0x6, a label switching router (LSR) misidentifies the Ethernet PW packet as an IP packet. The router considers that there is an IPv4 or IPv6 packet inside the MPLS packet and tries to load balance based on a hash of the source and destination IPv4 or IPv6 addresses extracted from the frame. This leads to the selection of the wrong equal-cost multipath (ECMP) path for the packet, leading to the misordering of packets.

This must not apply to an Ethernet frame that is encapsulated and transported over a PW because the destination MAC address considers the bottom label.

To overcome this issue, use the **control-word** keyword under a pw-class that is attached to a point-to-point PW. The control word is inserted immediately after the MPLS labels. Pseudowire over MPLS also, known as Ethernet over MPLS (EoMPLS), allows you to tunnel two L2VPN Provider Edge (PE) devices to transport L2VPN traffic over an MPLS cloud. This feature uses MPLS labels to transport data over the PW. The two L2VPN PEs are typically connected at two different sites with an MPLS core between them. This feature allows you to migrate legacy ATM and Frame Relay services to MPLS or IP core without interrupting the existing services.

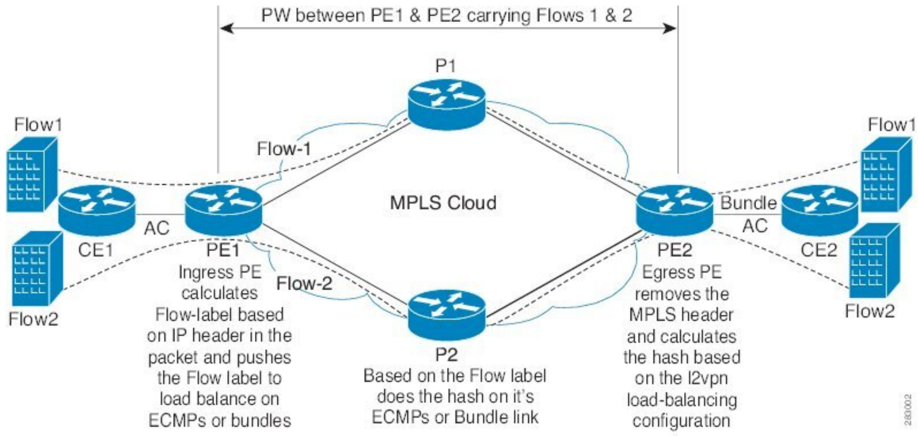
Load balancing using Flow-Label

Routers typically load balance traffic based on the lowermost label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric load balancing. 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.

The flow-label provides the capability to identify individual flows within a pseudowire and provides routers the ability to use these flows to load balance traffic. A flow-label is created based on individual packet flows entering a pseudowire and is inserted as the lowermost label in the packet. Routers can use the flow-label for load balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.

Topology

This example illustrates two flows distributing over ECMPs and bundle links.



Configure Load balancing using Control-Word and Flow-Label

Perform this task to configure load balancing using the **control-word** and **flow-label**.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class path1
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc)# control-word
Router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
Router(config-l2vpn-pwc-mpls)# exit
Router(config-l2vpn-pwc)# exit
Router(config-l2vpn)# xconnect group grp1
Router(config-l2vpn-xc)# p2p vlan1
Router(config-l2vpn-xc-p2p)# interface HundredGigE0/0/0/1.2
Router(config-l2vpn-xc-p2p)# neighbor 10.0.0.2 pw-id 2000
Router(config-l2vpn-xc-p2p-pw)# pw-class path1
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows the running configuration.

```
l2vpn
 pw-class path1
  encapsulation mpls
   control-word
   load-balancing
    flow-label both
  !
 !
 xconnect group grp1
  p2p vlan1
   interface HundredGigE0/0/0/1.2
   neighbor ipv4 10.0.0.2 pw-id 2000
    pw-class path1
   !
```

L2VPN Traffic Load Balancing on PE Router

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
Introduced New VLAN Tag Format Support for Load Balancing	Release 24.3.1	Introduced in this release on: Fixed Systems (8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Introduced support for the following VLAN tags on line cards and routers with Q100, Q200, and P100 based Silicon One ASIC: • Single VLAN tag 0x88A8 • QinQ with outer 0x8100 and inner 0x8100 • QinQ with outer 0x9100 and inner 0x8100 Introduced support for BUM traffic in VPLS service load balancing.
Extended L2VPN Traffic Load Balancing Support for line cards and routers with the P100 based Silicon One ASIC	Release 24.1.1	Introduced support for the following VLAN tags on line cards and routers with P100 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ outer 0x88A8 and inner 0x8100 Introduced support for QinQ with outer 0x8100 and inner 0x8100 on line cards and routers with Q200 based Silicon One ASIC.
Extended L2VPN Traffic Load Balancing Support for line cards and routers with the Q200 based Silicon One ASIC	Release 7.5.1	Introduced support for the following VLAN tags on line cards and routers with Q200 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ outer 0x88A8 and inner 0x8100

Feature Name	Release Information	Feature Description
L2VPN Traffic Load Balancing on PE Router	Release 3.7.1	Distributes L2 VPN traffic across multiple physical links or paths. Introduced support for the following VLAN tags on line cards and routers with Q100 based Silicon One ASIC: • Single VLAN tag 0x8100 • QinQ with outer 0x88A8 and inner 0x8100

A Provider Edge (PE) router balances Layer 2 Virtual Private Network (L2VPN) traffic by efficiently distributing it across network paths using various load balancing techniques.

On Cisco 8000, different load balance methods are used for different traffic groups:

1. Traffic in VPWS (E-Line) Service - For more information, see VPWS Service and Unicast Traffic in VPLS Service Load Balancing, on page 48
2. Unicast Traffic in VPLS (E-LAN) Service - For more information, see VPWS Service and Unicast Traffic in VPLS Service Load Balancing, on page 48.
3. Broadcast, Unknown Unicast, and Multicast (BUM) Traffic in VPLS (E-LAN) Service - For more information, see BUM Traffic in VPLS Service Load Balancing, on page 48.

Load balance is orthorgnal to point-to-point or multi-point connectivity. Load balance is needed over

1. Equal-Cost Multi-Path (ECMP) paths
2. Link Aggregation (LAG) bundle members

Supported VLAN Tag Formats for Load Balancing

The Cisco 8000 load balances traffic when it is either not VLAN tagged or tagged with VLAN in supported formats.

Table 11: Supported VLAN Tag Formats

VLAN Format	Q100 supports VLAN from Release	Q200 supports VLAN from Release	P100 supports VLAN from Release	A100 supports VLAN from Release
Single VLAN Tag 0x8100	3.7.1	7.5.1	24.1.1	24.1.1
Single VLAN Tag 0x88A8	24.3.1	24.3.1	24.3.1	24.3.1
QinQ outer 0x88A8, inner 0x8100 (double VLAN tags)	3.7.1	7.5.1	24.1.1	24.1.1

VLAN Format	Q100 supports VLAN from Release	Q200 supports VLAN from Release	P100 supports VLAN from Release	A100 supports VLAN from Release
QinQ outer 0x8100, inner 0x8100 (double VLAN tags)	24.3.1	24.1.1	24.3.1	24.3.1
QinQ outer 0x9100, inner 0x8100 (double VLAN tags)	24.3.1	24.3.1	24.3.1	24.3.1

VPWS Service and Unicast Traffic in VPLS Service Load Balancing

The router performs load balancing on outgoing interfaces and bundle members in these scenarios:

1. Ethernet frames entering from a L2 main or sub interface may be
    - switched out to another L2 interface that is part of a bundle, or
    - routed out to L3 interfaces with MPLS encapsulation.
2. MPLS labeled PW and EVPN traffic entering from an L3 interface. After label disposition, the customer Ethernet frames may be
    - switched out to an L2 main or sub interface that is part of a bundle, or
    - routed out to L3 interfaces with MPLS encapsulation.

The router parses packets to identify the required headers for generating a load balance hash, which determines the path to route the traffic across the network. The hashing process varies based on whether the traffic is received on L2 or L3 interfaces.

For load balance hash on traffic received from L2 interfaces, refer to Hash for Traffic Received on L2 Interface, on page 49.

For load balance hash on traffic received from L3 interfaces, refer to Hash for Traffic Received on L3 Interface, on page 50.

BUM Traffic in VPLS Service Load Balancing

The router performs BUM Traffic load balancing on outgoing interfaces and bundle members in these scenarios:

- Sending Traffic to an L2 Interface on bundle
- Sending Traffic over an MPLS PW
- Sending Traffic to EVPN MPLS Network

Sending Traffic to an L2 Interface on Bundle

When sending traffic to a L2 interface, the router does not perform load balancing over bundle members. Instead, it pins all traffic sent to an L2 main or sub-interface to a single bundle member. The selection of the bundle member is based on the main or sub-interface ID.

Sending Traffic over an MPLS PW

When BUM traffic is routed to an MPLS PW, the traffic is routed out to L3 interfaces. The router performs ECMP and bundle load balancing on the PW traffic. The hashing for load balancing is based on:

- PW VC label and flow label
- The 12 bytes of the PW payload. If control word (CW) is enabled, this includes a combination of 4 bytes of CW and 8 bytes of inner Ethernet MAC address.

Sending Traffic to EVPN MPLS Network

When BUM traffic is routed to an EVPN MPLS network, the traffic is encapsulated with:

- EVPN label
- ETREE leaf label or Ethernet Segment split horizon label

The MPLS encapsulated traffic is routed out to L3 interfaces. ECMP and bundle load balancing are performed on the EVPN MPLS encapsulated traffic. The hashing for load balancing is based on:

- EVPN label
- ETREE leaf label or Ethernet Segment split horizon label
- The 12 bytes of the PW payload. If CW is enabled, this includes a combination of 4 bytes of CW and 8 bytes of inner Ethernet MAC address.

MPLS non-IP Hash Disabled Configuration

In certain configurations, you may choose to disable the non-IP hash mode. When non-IP hash mode is disabled, the Ethernet MAC address of BUM traffic isn't used to perform load balance hashing. This can impact how BUM traffic is distributed across the network, as the router relies on other available headers for hashing.

Hash for Traffic Received on L2 Interface

For traffic received on a L2 interface, hashing depends on the headers present in the packet stack. The router generates the load balance hash using the designated fields based on the packet stack headers.

When ethernet frames entering a L2 main or sub interface, the router identifies the IP header within the packet based on the packet stack headers. The router uses these packet stack headers to generate the hash for traffic received on the L2 interface.

IP Traffic on L2 Interface

An ethernet frame is classified as IP traffic if it meets the following requirements:

- The ethernet header contains no more than two VLAN tags.
- The ethernet header either has no VLAN tag or has VLAN tags in a supported format as listed in Supported VLAN Tag Formats for Load Balancing, on page 47.
- No more than ten MPLS labels are placed in front of the IP header.

Non-IP Traffic on L2 Interface

All traffic that does not meet the description of IP traffic on L2 Interface is classified as non-IP traffic on L2 interface.

L2 Hash Fields

The L2 hash fields include the source and destination MAC addresses and the outer VLAN ID.

L3 Hash Fields

The L3 hash fields include the source and destination IP addresses and the source and destination ports of the layer 4 header.

IP Traffic Load Balancing

1. On Q100 and Q200 based Silicon One ASIC:

   Before Release 24.2.1, L2 hash fields were used in hashing. L3 hash fields were also included for limited traffic types.

   Starting from the Release 24.2.1, both L2 and L3 fields are used in hashing for all IP traffic.

2. On P100 and A100 based Silicon One ASIC:

   L2 hash fields are used in hashing. L3 hash fields are also included for limited traffic types.

Non-IP Traffic Load Balancing

Non-IP traffic on the L2 interface is load balanced using L2 hash fields.

If any of the designated fields are missing, the router replaces those field values with zeros.

Hash for Traffic Received on L3 Interface

For traffic received on a L3 interface hashing depends on several criteria, including the headers present in the packet stack and whether the Control Word (CW) and Flow Label (FL) are enabled on the pseudowire (PW).

When ethernet frames entering a L3 interface, the router identifies the IP header within the packet based on the packet stack headers.

IP over PW Traffic

An ethernet frame is classified as IP over PW traffic if it meets the following criteria:

- The frame contains an outer ethernet header and an inner ethernet header.
- No more than two MPLS labels are placed between the outer ethernet header and the inner ethernet header.
- There is no control word in front of the inner ethernet header.
- The inner ethernet header contains no more than two VLAN tags.
- The inner ethernet header has no VLAN tag or has VLAN tags in supported format as listed in Supported VLAN Tag Formats for Load Balancing, on page 47.
- Behind the inner ethernet header, there are no more than ten MPLS labels placed in front of the IP header.

Non-IP over PW Traffic

All traffic that does not meet the IP over PW traffic criteria is classified as non-IP over PW traffic.

Inner Ethernet L2 Hash Fields

The inner ethernet L2 hash fields include the source and destination MAC addresses, and the first VLAN tag of the inner ethernet frame.

Inner Ethernet L3 Hash Fields

The inner ethernet L3 hash fields include L3 and L4 headers in the inner ethernet frame. They are source and destination IP addresses, the source and destination ports of the layer 4 header.

Control Word Presence L2 Hash Fields

The 12 bytes of the PW payload, which include 4 bytes of CW followed by the 8 bytes of inner ethernet frame MAC address.

MPLS Hash Fields of Outer Ethernet Frame

All MPLS labels in front of inner ethernet header.

PW Disposition Traffic Load Balancing

1. PW disposition traffic load balance when control word and flow label are both disabled.

   IP over PW Traffic Load Balancing

   a. On Q100 and Q200 based Silicon One ASIC:

      IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

      Before Release 24.2.1, inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

      Starting from Release 24.2.1, both inner ethernet L2 hash fields and inner ethernet L3 hash fields are used in hashing.

   b. On P100 and A100 based Silicon One ASIC:

      IP over PW traffic load balance hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

   Non-IP over PW Traffic Load Balancing

   Non-IP over PW traffic load balance hash always uses the inner ethernet L2 hash fields.

2. PW disposition traffic load balance when control word is enabled and flow label disabled.

   In this case, all PW traffic is non-IP over PW traffic. Load balance hash uses control word presence L2 hash fields.

3. PW disposition traffic load balance when control word is disabled and flow label enabled.

   a. On Q100 and Q200 based Silicon One ASIC:

      IP over PW Traffic Load Balancing

      IP over PW traffic load balancing hash uses the following fields:

      - Before Release 24.2.1, hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.
      - In Release 24.2.1, inner ethernet L2 hash fields and inner ethernet L3 hash fields are both used in hashing.

- Starting from Release 24.3.1, inner ethernet L3 hash fields and MPLS hash fields of outer ethernet frame are used in hashing.

Non-IP over PW Traffic Load Balancing

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Starting from Release 24.3.1 release, MPLS hash fields of outer ethernet frame are used in hashing.

b. On P100 and A100 based Silicon One ASIC:

IP over PW Traffic Load Balancing

IP over PW traffic load balance hash uses the following fields:

- Before Release 24.3.1, hash uses the inner ethernet L2 hash fields. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.
- Starting from Release 24.3.1, hash uses MPLS hash fields of outer ethernet frame. Inner ethernet L3 hash fields are also added in the hashing for limited types of traffic.

Non-IP over PW Traffic Load Balancing

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the inner ethernet L2 hash fields.

Starting from Release 24.3.1 release, MPLS hash fields of outer ethernet frame are used in hashing.

4. PW disposition traffic load balance when control word and flow label are both enabled.

In this case, all PW traffic is Non-IP over PW traffic.

Before Release 24.3.1, Non-IP over PW traffic load balance hash uses the control word presence L2 hash fields.

Starting from Release 24.3.1, MPLS hash fields of outer ethernet frame are used in hashing.

G.8032 Ethernet Ring Protection Switching

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
G.8032 Ethernet Ring Protection Switching	Release 24.2.11	Ethernet Ring Protection Switching (ERPS) protocol, defined in ITU-T G.8032, provides protection for Ethernet traffic in a ring topology, while ensuring that there are no loops within the ring at the Ethernet layer. The loops are prevented by blocking either a predetermined link or a failed link. This feature introduces the **ethernet ring g8032** and **ethernet ring g8032 profile** commands.

ERPS ensures that link or node failures recover faster in Ethernet ring topologies. During a link failure, it reroutes traffic to provide continuous connectivity, simplifies network management, and operates independently of the control planes.

Overview

Each Ethernet ring node is connected to adjacent Ethernet ring nodes participating in the Ethernet ring using two independent links. A ring link never allows the formation of loops that affect the network. The Ethernet ring uses a specific link to protect the entire Ethernet ring. This specific link is called the ring protection link (RPL). A ring link is bound by two adjacent Ethernet ring nodes and a port for a ring link (also known as a ring port).

Note The minimum number of Ethernet ring nodes in an Ethernet ring is two.

The fundamentals of ring protection switching are:

- The principle of loop avoidance
- The utilization of learning, forwarding, and Filtering Database (FDB) mechanisms

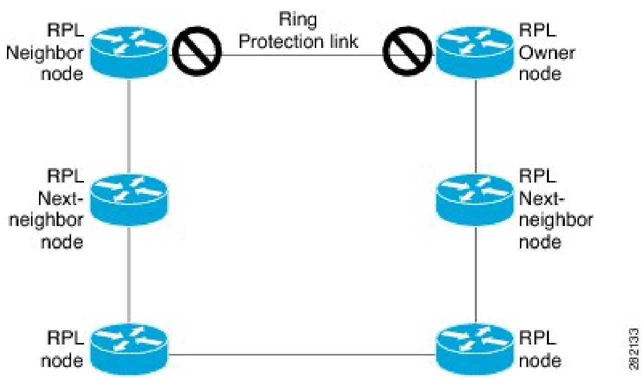
Loop avoidance in an Ethernet ring is achieved by ensuring that, at any time, traffic flows on all but one of the ring links which is the RPL. Multiple nodes are used to form a ring:

- RPL owner - It’s responsible for blocking traffic over the RPL so that no loops are formed in the Ethernet traffic. There can be only one RPL owner in a ring.
- RPL neighbor node - The RPL neighbor node is an Ethernet ring node next to the RPL. It's responsible for blocking its end of the RPL under normal conditions. This node type is optional and prevents RPL usage when protected.

- RPL next-neighbor node - The RPL next-neighbor node is an Ethernet ring node next to the RPL owner node or RPL neighbor node. It's used for FDB flush optimization on the ring. This node is also optional.

The following figure illustrates the G.8032 Ethernet ring.

Figure 10: G.8032 Ethernet Ring



Nodes on the ring use control messages called RAPS to coordinate the activities of switching on or off the RPL link. Any failure along the ring triggers a RAPS signal fail (RAPS SF) message along both directions, from the nodes next to the failed link, after the nodes have blocked the port facing the failed link. On obtaining this message, the RPL owner unblocks the RPL port.

Note A single link failure in the ring ensures a loop-free topology.

Line status and Connectivity Fault Management protocols are used to detect ring link and node failure. During the recovery phase, when the failed link is restored, the nodes next to the restored link send RAPS no request (RAPS NR) messages. On obtaining this message, the RPL owner blocks the RPL port and sends RAPS no request, root blocked (RAPS NR, RB) messages. This causes all other nodes, other than the RPL owner in the ring, to unblock all blocked ports. The ERPS protocol is robust enough to work for both unidirectional failure and multiple link failure scenarios in a ring topology.

A G.8032 ring supports these basic operator administrative commands:

- Force switch (FS) - Allows the operator to forcefully block a particular ring-port.
  - Effective even if there’s an existing SF condition.
  - Multiple FS commands for ring supported.
  - May be used to allow immediate maintenance operations.
- Manual switch (MS) - Allows the operator to manually block a particular ring-port.
  - Ineffective in an existing FS or SF condition.
  - Overridden by new FS or SF conditions.
  - Multiple MS commands cancel all MS commands.
- Clear - Cancels an existing FS or MS command on the ring-port.

- Used (at RPL Owner) to clear non-revertive mode.

A G.8032 ring can support multiple instances. An instance is a logical ring running over a physical ring. Such instances are used for various reasons, such as load balancing VLANs over a ring. For example, odd VLANs may go in one direction of the ring, and even VLANs may go in the other direction. Specific VLANs can be configured under only one instance. They cannot overlap multiple instances. Otherwise, data traffic or RAPS packets can cross logical rings, and that isn't desirable.

G.8032 ERPS provides a new technology that relies on line status and Connectivity Fault Management (CFM) to detect link failure. By running CFM Continuity Check Messages (CCM) messages at an interval of 100ms, it's possible to achieve SONET-like switching time performance and loop free traffic.

For more information about Ethernet Connectivity Fault Management (CFM) and Ethernet Fault Detection (EFD) configuration, refer to the *Configuring Ethernet OAM on the Cisco 8000 Series Router* module in the *Cisco 8000 Series Router Component Configuration Guide*.

Timers

G.8032 ERPS specifies the use of different timers to avoid race conditions and unnecessary switching operations:

- Delay Timers - used by the RPL Owner to verify that the network has stabilized before blocking the RPL.
    - After SF condition, a Wait-to-Restore (WTR) timer is used to verify that SF isn't intermittent. The WTR timer can be configured by the operator, and the default time interval is 5 minutes. The time interval ranges 1–12 minutes.
    - After the FS/MS command, a Wait-to-Block timer is used to verify that no background condition exists.

Note The Wait-to-Block timer may be shorter than the Wait-to-Restore timer.

- Guard Timer - used by all nodes when changing state; it blocks latent outdated messages from causing unnecessary state changes. The Guard timer can be configured and the default time interval is 500 ms. The time interval ranges 10-2000 ms.
- Hold-off timers - used by the underlying Ethernet layer to filter out intermittent link faults. The hold-off timer can be configured and the default time interval is 0 seconds. The time interval ranges 0–10 seconds.
    - Faults are reported to the ring protection mechanism, only if this timer expires.

During a link failure, the G8032 EPR performs either of the following operations to provide continuous connectivity:

- If it's unable to recover from the link failure, it reroutes traffic. For more information, refer to Protection Switching during Single Link Failure, on page 56.
- Wait to recover from link failure to prevent unnecessary switching operations. For more information, refer to Recovery from Single Link Failure, on page 57.

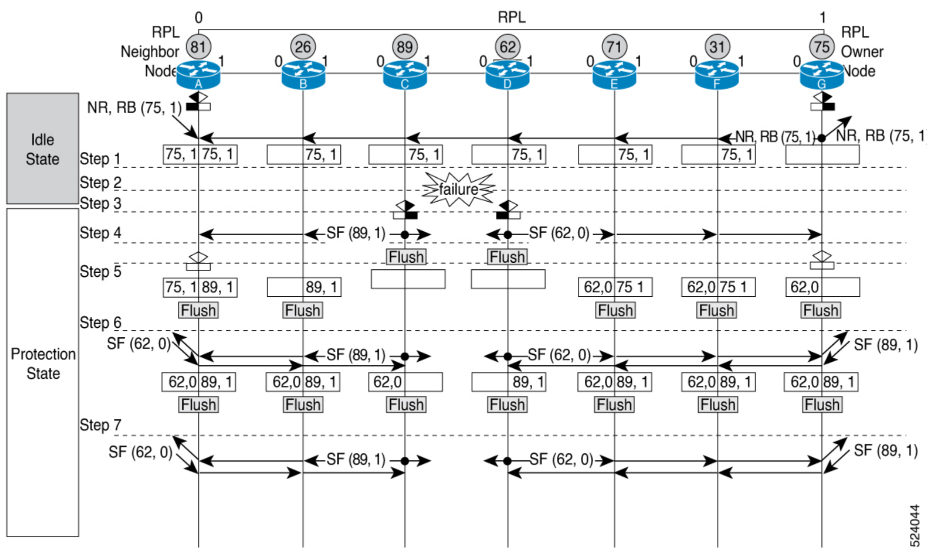
Protection Switching during Single Link Failure

The following example describes the protection switching process during a single link failure:

For example, consider the Figure 11: Protection Switching during G.8032 Single Link Failure, on page 56 figure with the following configuration:

- An ethernet ring is composed of seven ethernet ring nodes (A to G) with node ID (81, 26, 89, 62, 71, 31, and 75) and port ID (0 and 1).
- The ethernet ring node A is the RPL neighbor node.
- The ethernet ring node G is the RPL owner node.
- The RPL is the ring link between ethernet ring nodes A and G.
- Traffic is blocked at both ends of the RPL.

Figure 11: Protection Switching during G.8032 Single Link Failure



The ERPS performs the following protection switching steps during a single link failure:

1. The link operates in the normal condition.
2. A failure occurs between ring nodes C and D.
3. Ethernet ring nodes C and D detect a local signal failure (SF) condition and after the holdoff time interval, block the failed ring port and perform the forwarding database (FDB) flush.
4. Ethernet ring nodes C and D start sending ring automatic protection switching (RAPS) (SF) messages periodically along with the node ID and Blocked Port Ring (BPR) pair on both ring ports, while the SF condition persists.

   For example, ring node C sends the SF(89,1) message, which consists of node ID 89 and BPR 1.
5. All Ethernet ring nodes receiving an RAPS (SF) message perform FDB flush. When the RPL owner node G and RPL neighbor node A receive an RAPS (SF) message, the Ethernet ring node unblocks its end of the RPL and performs the FDB flush.

6. All Ethernet ring nodes receiving a second RAPS (SF) message perform the FDB flush again; this is because of the Node ID and BPR-based mechanism.
7. Stable SF condition—RAPS (SF) messages on the Ethernet Ring. Further RAPS (SF) messages trigger no further action.

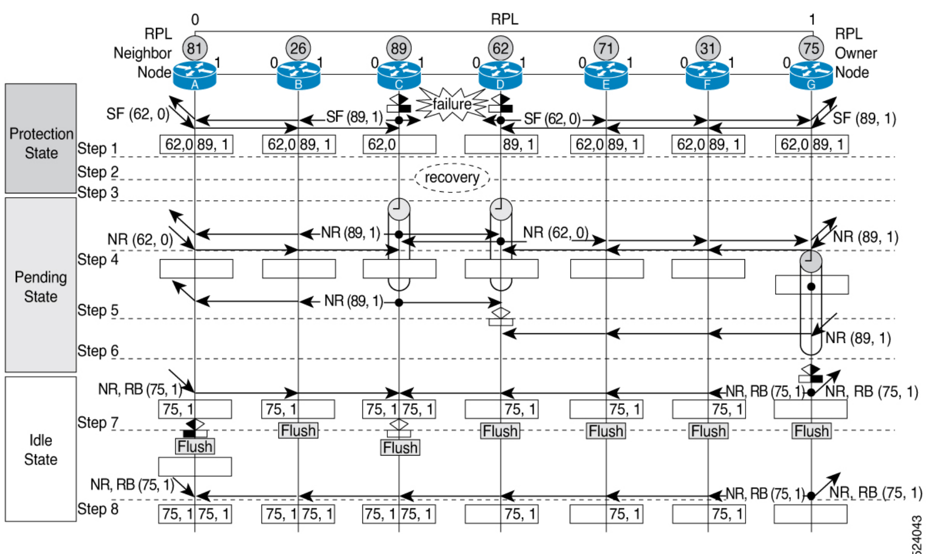
Recovery from Single Link Failure

The following example describes the single link failure recovery process:

For example, consider the Figure 12: Single link failure Recovery (Revertive operation), on page 57 figure with the following configuration:

- An ethernet ring is composed of seven ethernet ring nodes (A to G) with node ID (81, 26, 89, 62, 71, 31, and 75) and port ID (0 and 1).
- The ethernet ring node A is the RPL neighbor node.
- The ethernet ring node G is the RPL owner node.
- The RPL is the ring link between ethernet ring nodes A and G.
- Traffic is blocked at both ends of the RPL.

Figure 12: Single link failure Recovery (Revertive operation)



The ERPS performs the following reversion steps to revertive the link during a single link failure:

1. A failure occurs between ring nodes C and D.
2. Recovery of link failure occurs between ring nodes C and D.
3. Ethernet ring nodes C and D detect clearing of SF condition, start the guard timer and initiate periodical transmission of RAPS No Request (NR) messages on both ring ports.

> **Note** The guard timer prevents the reception of RAPS messages.

4. When the Ethernet ring nodes receive an RAPS (NR) message, the node ID and BPR pair of a receiving ring port is deleted and the RPL owner node starts the WTR timer.
5. When the guard timer expires on ethernet ring nodes C and D, they may accept the new RAPS messages that they receive. Ethernet ring node D receives an RAPS (NR) message with higher Node ID from ethernet ring node C, and unblocks its non-failed ring port.
6. When the WTR timer expires, the RPL owner node blocks its end of the RPL, sends the RAPS (NR, RB) message with the node ID and BPR pair, and performs the FDB flush.
7. When Ethernet ring node C receives an RAPS (NR, RB) message, it removes the block on its blocked ring ports, and stops sending RAPS (NR) messages. On the other hand, when the RPL neighbor node A receives an RAPS (NR, RB) message, it blocks its end of the RPL. In addition to this, Ethernet ring nodes A to F perform the FDB flush when receiving an RAPS (NR, RB) message, due to the existence of the Node ID and BPR based mechanism.
8. Link operates in the stable SF condition.

Restrictions for G.8032 Ethernet Ring Protection Switching

- You must not configure G.8032 ERPS and CFM down-mep on the same sub-interface. If you enable it, then the router displays a syslog message, as shown in the following example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# ethernet ring g8032 test
Router(config-l2vpn-erp)# port0 interface FourHundredGigE0/0/0/6
Router(config-l2vpn-erp)# port1 interface FourHundredGigE0/0/0/10
Router(config-l2vpn-erp)# instance 1
Router(config-l2vpn-erp-inst)# profile test
Router(config-l2vpn-erp-inst)# rpl port0 owner
Router(config-l2vpn-erp-inst)# inclusion-list vlan-ids 1,100
Router(config-l2vpn-erp-inst)# aps-channel
Router(config-l2vpn-erp-inst-aps)# port0 interface FourHundredGigE0/0/0/6.1
Router(config-l2vpn-erp-inst-aps)# port1 interface FourHundredGigE0/0/0/10.1

Router(config)# interface FourHundredGigE0/0/0/6.1 l2transport
Router(config-if)# encapsulation dot1q 1
Router(config-if)# ethernet cfm
Router(config-if-cfm)# mep domain domain1 service link1 mep-id 2

%PLATFORM-SPITFIRE_CFM-3-G8032_VIOLATION : G8032 has been configured for interface
FourHundredGigE0/0/0/6.1
where CFM configuration exists. G8032 config is disabled.
```

  Instead configure the CFM down-mep on the main interface and configure the G.8032 ERPS on the sub-interface.

- Linecards and fixed routers with Q100 and Q200 based Silicon One ASICs don't support G.8032 Ethernet Ring Protection Switching.

Configuring G.8032 Ethernet Ring Protection Switching

To configure the G.8032 operation, you have to configure ERPS and CFM separately as follows:

- Configure the ERPS profile, ERPS instance, ERPS paramenters, and TCN propagation by including the following requirements:
    - Designate a (sub)interface which is used as the APS channel.
    - Designate a (sub)interface which is monitored by CFM.
    - Verify whether the interface is an RPL link, and, if it is a RPL link then indicate the RPL node type.

  For more information, see the following sections:
    - Configuring ERPS Profile, on page 59
    - Configuring an ERPS Instance, on page 60
    - Configuring ERPS Parameters, on page 62
    - Configuring TCN Propagation, on page 63
- Configure CFM with EFD to monitor the ring links. For more information, see the Configuring CFM MEP, on page 63.

Note MEP for each monitor link needs to be configured with different Maintenance Association.

- The bridge domains to create the Layer 2 topology. The RAPS channel is configured in a dedicated management bridge domain separated from the data bridge domains.
- Behavior characteristics, that apply to ERPS instance, if different from default values. This is optional.

This section provides information on:

Configuring ERPS Profile

Perform this task to configure the Ethernet Ring Protection Switching (ERPS) profile.

Step 1 Configure a new G.8032 ERPS profile using the **Ethernet ring g8032 profile** command.

Example:

```
Router# configure
Router(config)# Ethernet ring g8032 profile p1
```

Enables G.8032 ring mode, and enters G.8032 configuration submode.

Step 2 Sets the hold-off timer using the **timer** command.

Example:

```
Router(config-g8032-ring-profile)# timer hold-off 5
```

Specifies a time interval (in seconds) for the guard, hold-off, and wait-to-restore timers.

The hold-off timer prevents unnecessary switching due to short-lived failures on the ring.

Step 3 Specify a non-revertive ring instance using the **non-revertive** command.

Example:

```
Router(config-g8032-ring-profile)# non-revertive
Router(config-g8032-ring-profile)# commit
```

This feature enables the router to use the current path until an administrator manually reverts to the original path.

Configuring an ERPS Instance

Perform this task to configure an ERPS instance.

Step 1 Configure the layer 2 VPN with a bridge group.

Example:

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

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

Step 2 Configure a bridge domain for R-APS channels using the **bridge-domain** command.

Example:

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

Establishes a bridge domain for R-APS channels, and enters L2VPN bridge group bridge domain configuration mode.

Step 3 Configure an interface to a bridge domain on ports 0 and 1 using the **interface** command.

Example:

```
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
```

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 4 Configure a bridge domain for data traffic using the **bridge-domain** command.

Example:

```
Router(config-l2vpn-bg)# bridge-domain bd2
```

Establishes a bridge domain for data traffic, and enters L2VPN bridge group bridge domain configuration mode.

Step 5 Configure an interface to a bridge domain using the **interface** command.

Example:

```
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.10
```

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 Configure an ethernet ring using the **ethernet ring g8032** command.

Example:

```
Router(config-l2vpn)# ethernet ring g8032 r1
```

Enables G.8032 ring mode, and enters G.8032 configuration submode.

Step 7 Configure an ERPS instance using the **instance** command.

Example:

```
Router(config-l2vpn-erp)# instance 1
```

Enters the Ethernet ring G.8032 instance configuration submode.

Step 8 Add a description for the ERPS instance that you are configuring using the **description** command.

Example:

```
Router(config-l2vpn-erp-instance)# description test
```

Specifies a string that serves as a description for that instance.

Step 9 Configure an ERPS profile using the **profile** command.

Example:

```
Router(config-l2vpn-erp-instance)# profile p1
```

Specifies associated Ethernet ring G.8032 profile.

Step 10 Specify the RPL port and designates it as a neighbor using the **rpl** command.

Example:

```
Router(config-l2vpn-erp-instance)# rpl port0 neighbor
```

Specifies one ring port on the local node as RPL owner, neighbor, or next-neighbor.

Step 11 Configure the VLANs that are included in the ERPS instance using the **inclusion-list vlan-ids** command.

Example:

```
Router(config-l2vpn-erp-instance)# inclusion-list vlan-ids e-g
```

Associates a set of VLAN IDs with the current instance.

Step 12 Enable Automatic Protection Switching (APS) channel configuration mode for the ERPS instance and sets the priority level for the APS protocol.

Example:

```
Router(config-l2vpn-erp-instance)# aps-channel
Router(config-l2vpn-erp-instance-aps)# level 5
```

Enters the ethernet ring G.8032 instance aps-channel configuration submode and specifies the APS message level. The range is 0–7.

Step 13 Assign a port to the G.8032 APS channel interface.

Example:

```
Router(configl2vpn-erp-instance-aps)# port0 interface GigabitEthernet 0/0/0/0.1
```

Associates G.8032 APS channel interface to port0.

Step 14 Assign a port to the G.8032 APS channel interface.

Example:

```
Router(config-l2vpn-erp-instance-aps)# port1 interface GigabitEthernet 0/0/0/1.1
```

Associates G.8032 APS channel interface to port1.

Configuring ERPS Parameters

Perform this task to configure ERPS parameters.

Step 1 Configure an ethernet ring in L2VPN configuration mode.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# ethernet ring g8032 r1
```

Enters L2VPN configuration mode, enables G.8032 ring mode, and enters G.8032 configuration submode.

Step 2 Enable G.8032 ERPS for the specified port (ring port).

Example:

```
Router(config-l2vpn-erp)# port0 interface GigabitEthernet 0/0/0/0
```

Step 3 Specify a port to monitor the G.8032 ERPS and detect ring link failure.

Example:

```
Router(config-l2vpn-erp-port0)# monitor port0 interface 0/0/0/0.5
```

Specifies the port that is monitored to detect ring link failure per ring port. The monitored interface must be a sub-interface of the main interface.

Step 4 Exit from port configuration submode.

Example:

```
Router(config-l2vpn-erp-port0)# exit
```

Exits port0 configuration submode.

Step 5 Enable G.8032 ERPS for the specified port (ring port).

Example:

```
Router(config-l2vpn-erp)# port1 interface GigabitEthernet 0/0/0/1
```

Enables G.8032 ERPS for the specified port (ring port).

Step 6 Specify a port to monitor the G.8032 ERPS and detect ring link failure.

Example:

```
Router(config-l2vpn-erp-port1)# monitor port1 interface 0/0/0/1.5
```

Specifies the port that is monitored to detect ring link failure per ring port. The monitored interface must be a sub-interface of the main interface.

Step 7 Exit from port configuration submode.

Example:

```
Router(config-l2vpn-erp-port1)# exit
```

Exits port1 configuration submode.

Step 8 Configure a set of VLAN IDs that isn't protected by the Ethernet ring protection mechanism using the **exclusion-list vlan-ids** command.

Example:

```
Router(config-l2vpn-erp)# exclusion-list vlan-ids a-d
```

Step 9 Configure the ethernet ring G.8032 as an open ring.

Example:

```
Router(config-l2vpn-erp)# open-ring
```

Configuring TCN Propagation

Perform this task to configure topology change notification (TCN) propagation.

Enable TCN propagation in L2VPN configuration mode.

Example:

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# tcn-propagation
Router(config-l2vpn)# commit
```

Enters L2VPN configuration mode and allows TCN propagation from minor ring to major ring and from MSTP to G.8032.

Configuring CFM MEP

Configuring the CFM on the main interface and G.8032 ERPS on the sub-interfaces allows you to constantly monitor the Ethernet ring's status. If a link in the ring fails, the Ethernet Fault Detection (EFD) shuts down the affected port and reroutes the traffic through the new path.

Perform the following steps to configure the CFM MEP:

Step 1 Configure a CFM domain and service.

Example:

```
Router# configure
Router(config)# ethernet cfm
Router(config-cfm)# domain dom23to24 level 6
Router(config-cfm-domain)# service ser23to24 down-meps
```

Step 2 Configure the continuity checks.

Example:

```
Router(config-cfm-svc)# continuity-check interval 10s
```

Step 3 Configure a MEP crosscheck on the main interface to detect failures and reroute the traffic.

Example:

```
Router(config-cfm-svc)# mep crosscheck
Router(config-cfm-svc-xcheck)# mep-id 3
Router(config-cfm-svc-xcheck)# exit
```

Step 4 Configure Ethernet Fault Detection (EFD) to detect failures and reroute the traffic.

Example:

```
Router(config-cfm-svc)# efd
```

Step 5 Configure CFM MEP on the sub-interface.

Example:

```
Router# configure terminal
Router(config)# interface Gigabiteethernet0/0/0/0.5
Router(config-if)# ethernet cfm
Router(config-if-cfm)# mep domain dom23to24 service ser23to24 mep-id 4
```

For more information about Ethernet Connectivity Fault Management (CFM), refer to the *Configuring Ethernet OAM on the Cisco 8000 Series Router* module in the *Cisco 8000 Series Router Interface and Hardware Component Configuration Guide*.

Configuring G.8032 Ethernet Ring Protection Switching: Example

This sample configuration illustrates the elements that a complete G.8032 configuration includes:

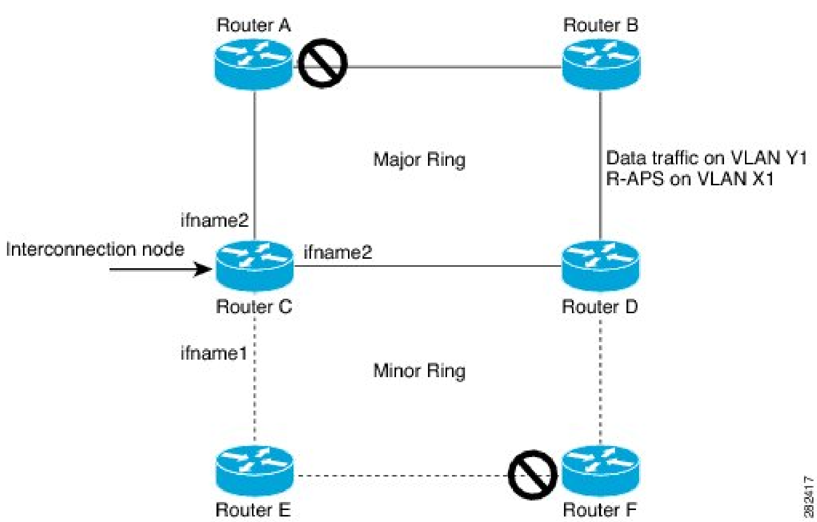
```
# Configure the ERP profile characteristics if ERPS instance behaviors are non-default.
ethernet ring g8032 profile ERP-profile
  timer wtr 60
  timer guard 100
  timer hold-off 1
  non-revertive


# Configure the ERPS instance under L2VPN
l2vpn
  ethernet ring g8032 RingA
    port0 interface g0/0/0/0
    port1 interface g0/1/0/0
    instance 1
      description BD2-ring
      profile ERP-profile
      rpl port0 owner
      vlan-ids 10-100
      aps channel
        level 3
        port0 interface g0/0/0/0.1
        port1 interface g1/1/0/0.1

# Set up the bridge domains
bridge group ABC
    bridge-domain BD2
      interface Gig 0/0/0/0.2
      interface Gig 0/1/0/0.2
      interface Gig 0/2/0/0.2

    bridge-domain BD2-APS
      interface Gig 0/0/0/0.1
      interface Gig 1/1/0/0.1
```

```
# EFPs configuration
interface Gig 0/0/0/0.1 l2transport
  encapsulation dot1q 5

interface Gig 1/1/0/0.1 l2transport
  encapsulation dot1q 5

interface g 0/0/0/0.2 l2transport
  encapsulation dot1q 10-100

interface g 0/1/0/0.2 l2transport
  encapsulation dot1q 10-100

interface g 0/2/0/0.2 l2transport
  encapsulation dot1q 10-100
```

Configuring Interconnection Node: Example

This example shows you how to configure an interconnection node. The following figure illustrates an open ring scenario.

Figure 13: Open Ring Scenario - interconnection node



The minimum configuration required for configuring G.8032 at Router C (Open ring – Router C):

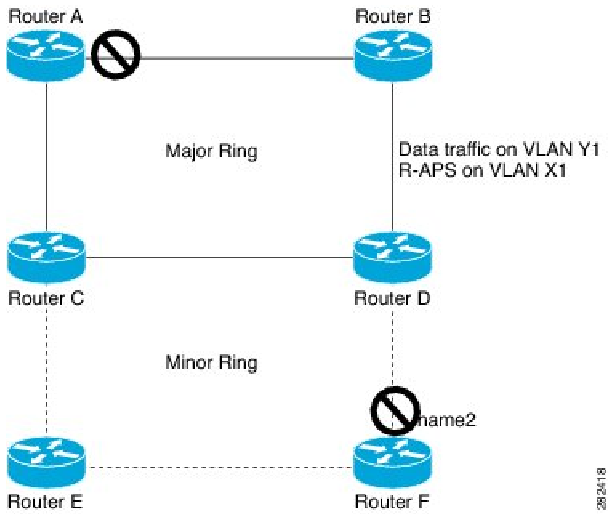
```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
interface <ifname3.10> l2transport
 encapsulation dot1q Y1
l2vpn
ethernet ring g8032 <ring-name>
      port0 interface <main port ifname1>
      port1 interface none #? This router is connected to an interconnection node
      open-ring #? Mandatory when a router is part of an open-ring
      instance <1-2>
         inclusion-list vlan-ids  X1-Y1
         aps-channel
           Port0 interface <ifname1.1>
           Port1 none #? This router is connected to an interconnection node
```

```
bridge group bg1
   bridge-domain bd-aps#? APS-channel has its own bridge domain
      <ifname1.1> #? There is only one APS-channel at the interconnection node
   bridge-domain bd-traffic #? Data traffic has its own bridge domain
      <ifname1.10>
      <ifname2.10>
      <ifname3.10>
```

Configuring the Node of an Open Ring: Example

This example shows you how to configure the node part of an open ring. The following figure illustrates an open ring scenario.

Figure 14: Open Ring Scenario



The minimum configuration required for configuring G.8032 at the node of the open ring (node part of the open ring at router F):

```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname2.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
l2vpn
   ethernet ring g8032 <ring-name>
      port0 interface <main port ifname1>
      port1 interface <main port ifname2>
      open-ring #? Mandatory when a router is part of an open-ring
      instance <1-2>
         inclusion-list vlan-ids  X1-Y1
      rpl port1 owner #? This node is RPL owner and <main port ifname2> is blocked
         aps-channel
           port0 interface <ifname1.1>
           port1 interface <ifname2.1>
 bridge group bg1
    bridge-domain bd-aps#? APS-channel has its own bridge domain
       <ifname1.1>
       <ifname2.1>
    bridge-domain bd-traffic #? Data traffic has its own bridge domain
```

```
<ifname1.10>
<ifname2.10>
```