L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers, IOS XR Release 7.6.x

First Published: 2022-03-30

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

From Release 6.1.2 onwards, Cisco introduces support for the 64-bit Linux-based IOS XR operating system. Extensive feature parity is maintained between the 32-bit and 64-bit environments. Unless explicitly marked otherwise, the contents of this document are applicable for both the environments. For more details on Cisco IOS XR 64 bit, refer to the Release Notes for Cisco ASR 9000 Series Routers, Release 6.1.2 document.

This guide describes the Cisco ASR 9000 Series Router configurations. The preface for the *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers* contains these sections:

Changes to This Document

The following table lists the technical changes made to this document since it was first published.

Date	**Change Summary**
March 2022	Initial release of this document.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the What's New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

CHAPTER 1

New and Changed VPN Features

This table summarizes the new and changed feature information for the L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers, and tells you where they are documented.

New and Changed VPN Features

Table 1: VPN Features Added or Modified in IOS XR Release 7.6.x

Feature	Description	Changed in Release	Where Documented
Configurable Recovery Time for EVPN Core Isolation Group	This feature was introduced.	Release 7.6.1	Configurable Recovery Time for EVPN Core Isolation Group, on page 629

CHAPTER 2

The Carrier Ethernet Model

This chapter introduces you to Layer 2 (L2) features and standards. This chapter also describes how to configure L2VPN features

The distributed Gigabit Ethernet and 10-Gigabit Ethernet architecture and features deliver network scalability and performance, while enabling service providers to offer high-density, high-bandwidth networking solutions designed to interconnect the router with other systems in POPs, including core and edge routers and L2 and Layer 3 (L3) switches.

Note This chapter does not include configuration information for Management Ethernet interfaces. To set up a Management Ethernet interface and enable Telnet servers, see the *Cisco ASR 9000 Series Aggregation Services Routers Getting Started Guide*. To configure a Management Ethernet interface for routing or to modify the configuration of a Management Ethernet interface, see the *Advanced Configuration and Modification of the Management Ethernet Interface on the Cisco ASR 9000 Series Router* chapter of the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

Feature History for Configuring Ethernet Interfaces

Release	Modification
Release 3.7.2	This feature was introduced on the Cisco ASR 9000 Series Routers.
Release 4.1.1	Scalability of EFPs on bundle interfaces was introduced.

Prerequisites for Configuring Layer 2 Ethernet Interfaces

Before configuring Ethernet interfaces, ensure that these tasks and conditions are met:

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

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

- Confirm that at least one of these line cards is installed on the router:
  - 4-port 10-Gigabit Ethernet (4 x 10 GE) line card
  - 8-port 10-Gigabit Ethernet (4 x 10 GE) line card
  - 40-port 1-Gigabit Ethernet line card
- You know the interface IP address.
- You know how to apply the specify the generalized interface name with the generalized notation *rack/slot/module/port* .

Layer 2 Theory and Standards Adherence

To configure Ethernet interfaces, you must understand these concepts:

Ethernet Technology Overview

Ethernet is defined by the IEEE 802.3 international standard. It enables the connection of up to 1024 nodes over coaxial, twisted-pair, or fiber-optic cable.

The Cisco ASR 9000 Series Routers support Gigabit Ethernet (1000 Mbps) and 10-Gigabit Ethernet (10 Gbps) interfaces.

Carrier Ethernet Services

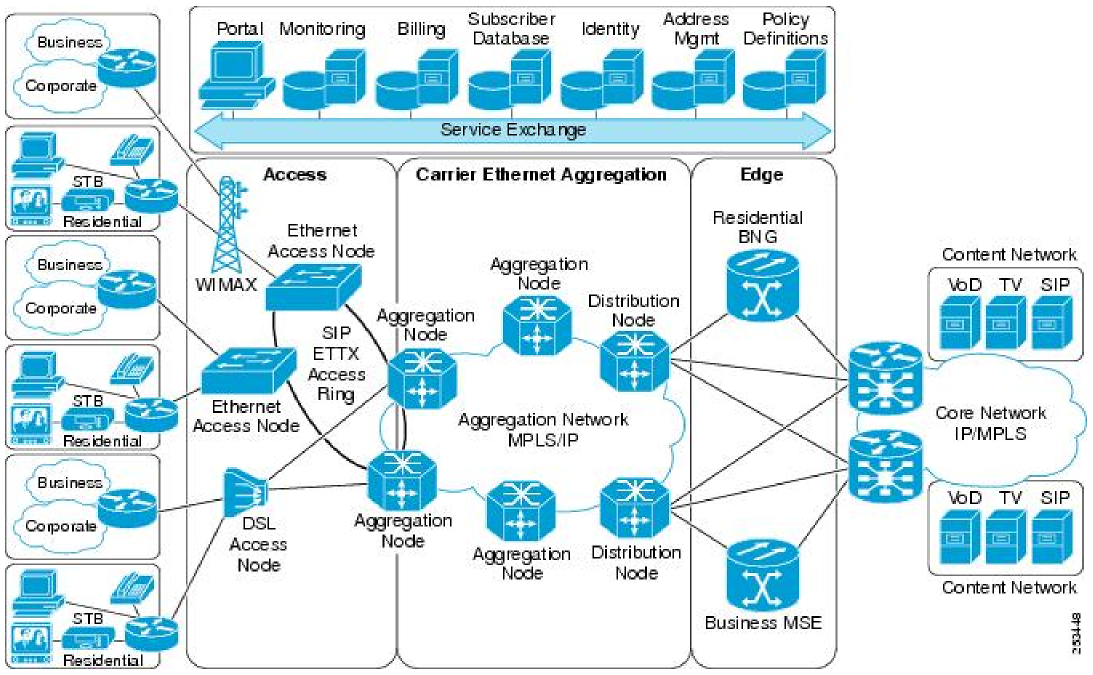
Cisco endorses Metro Ethernet Forum (MEF) Carrier Ethernet (CE) 2.0 services:

- E-LINE: An E-Line is a point-to-point Ethernet service that connects exactly 2 UNIs.
- E-LAN: An E-LAN is a multipoint-to-multipoint service that connects a number of UNIs (2 or more) providing full mesh connectivity for those sites.
- E-TREE: An E-Tree is a rooted multipoint service that connects a number of UNIs providing sites with hub and spoke multipoint connectivity.
- E-ACCESS: An E-Access Service is an OVC-based service with at least one UNI OVC End Point and one ENNI End Point.

When discussing an Ethernet WAN (EWAN), these terminologies should be used:

- CE (customer edge): The customer device connecting to the service provider
- PE (provider edge): The service provider device connecting to the customer
- UNI: The connection between the CE and PE
- AC: The physical or virtual circuit attaching a CE to a PE
- Multiplexed UNI: A UNI supporting multiple VLAN flows
- Pseudowire: A term used to indicate an end-to-end path in a service provider network

Figure 1: EWAN Terms



Ethernet Wire Service

An Ethernet Wire Service is a service that emulates a point-to-point Ethernet segment. This is similar to Ethernet private line (EPL), a Layer 1 point-to-point service, except the provider edge operates at Layer 2 and typically runs over a Layer 2 network. The EWS encapsulates all frames that are received on a particular UNI and transports these frames to a single-egress UNI without reference to the contents contained within the frame. The operation of this service means that an EWS can be used with VLAN-tagged frames. The VLAN tags are transparent to the EWS (bridge protocol data units [BPDUs])—with some exceptions. These exceptions include IEEE 802.1x, IEEE 802.2ad, and IEEE 802.3x, because these frames have local significance and it benefits both the customer and the Service Provider to terminate them locally.

Since the service provider simply accepts frames on an interface and transmits these without reference to the actual frame (other than verifying that the format and length are legal for the particular interface) the EWS is indifferent to VLAN tags that may be present within the customer Ethernet frames.

EWS subscribes to the concept of all-to-one bundling. That is, an EWS maps a port on one end to a point-to-point circuit and to a port on another end. EWS is a port-to-port service. Therefore, if a customer needs to connect a switch or router to n switches or routers it will need n ports and n pseudowires or logical circuits.

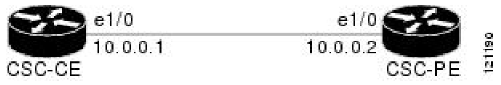
One important point to consider is that, although the EWS broadly emulates an Ethernet Layer 1 connection, the service is provided across a shared infrastructure, and therefore it is unlikely that the full interface bandwidth will be, or needs to be, available at all times. EWS will typically be a sub-line rate service, where many users share a circuit somewhere in their transmission path. As a result, the cost will most likely be less than that of EPL. Unlike a Layer 1 EPL, the SP will need to implement QoS and traffic engineering to meet the specific objectives of a particular contract. However, if the customer's application requires a true wire rate transparent service, then an EPL service—delivered using optical transmission devices such as DWDM (dense wavelength division multiplexing), CDWM (coarse wavelength division multiplexing), or SONET/SDH—should be considered.

Ethernet Virtual Private Line

Ethernet Virtual Private Line (EVPL) is similar to EWS in that it offers point-to-point connectivity. The key differentiation between EWS and EVPL is that an EVPL uses a VLAN tag to multiplex several, non-same-destination pseudowires to one port. That is, unlike EPL and EWS, EVPL is a one-to-many multiplexed service. Service multiplexing simply means that multiple pseudowires utilize a single access interface or UNI. These circuits can terminate within an L2VPN or on, for example, an Internet gateway. From the service user's perspective, this service multiplexing capability offers more efficient interface utilization, simplification of cable plant, and reduced maintenance costs associated with additional interfaces.

Using the same example as above, where a router connects to n other routers, the source router only needs one port for the service instead of n, as is the case with an EWS. The service need not be port-to-port, but can be logical-pseudowire-to-logical-pseudowire. In the case of an EVPL, each circuit can terminate at a different remote location, whereas using EWS, all frames are mapped to a single circuit and therefore a single egress point.

Figure 2: EVPL Service Multiplexing Example: One Port (Left) Can Be Used for All Destinations (Right)



Like Frame Relay, EVPL allows a customer device to access multiple connections through a single physical port attached to the service provider network. The service offered by EVPL can be thought of as being similar in concept to Frame Relay, in that a VLAN number is used as a virtual circuit identifier in a similar fashion to Frame Relay data link connection identifier (DLCI). Unlike EWS, EVPL does not forward BPDUs, because IEEE 802.1Q (VLAN tagging) only sends BPDUs on a default VLAN. In a hub-and-spoke network, only one spoke at most would receive BPDUs, thus breaking the spanning tree in the rest of the network. Therefore, an EVPL does not transmit any BPDUs and runs routing protocols instead of Ethernet Spanning Tree. The routing protocols give the customer and provider greater flexibility, traffic determination characteristics, and value-added services.

Ethernet LAN Service

An Ethernet LAN Service (E-LAN) differs from EWS and ERS in that an E-LAN provides a multipoint connectivity model. It should be noted that an E-LAN service definition is still under review within the IETF Multipoint Layer 2 Services working group. Although E-LAN uses a multipoint model, it can forward unicast packets to single destinations; that is, it also supports point-to-point connections. To the end user, the network looks like a giant Ethernet switch where each customer has their own VLAN or broadcast domain, rather than end-to-end pseudowire link(s).

E-LAN Example

An E-LAN does not map an interface or VLAN to a specific point-to-point pseudowire. Instead, it models the operation of a virtual Ethernet switch: E-LAN uses the customer's MAC address to forward frames to the correct egress UNI within the service provider's network. An E-LAN emulates the service attributes of an Ethernet switch and learns source MAC to interface associations, floods unknown broadcast and multicast frames, and (optionally) monitors the service user's spanning tree protocol. One important point to note is that although the service provider may utilize spanning tree within the transport network, there is no interaction with the service user's spanning tree.

This service works similar to an MPLS VPN, except it functions at L2 instead of L3. While a VPLS E-LAN is a viable solution, its scalability and QoS control are suspect compared to that of MPLS VPNs. In addition, it is much more difficult, and may be impossible, for the service provider to offer value-added Layer 3 services (this is discussed later in the document).

Ethernet Flow Point

An Ethernet Flow Point (EFP) is a substream partition of a main interface. On Cisco ASR 9000 Series Routers, the EFP is implemented as an L2 subinterface with an encapsulation statement.

Ethernet Virtual Circuit

An Ethernet Virtual Circuit (EVC) is a point-to-point tunnel. On Cisco ASR 9000 Series Routers, the EVC is implemented as a pseudowire (PW).

Ethernet OAM Protocols

Ethernet as a Metro Area Network (MAN) or a Wide Area Network (WAN) technology benefits greatly from the implementation of Operations, Administration and Maintenance (OAM) features. OAM features allow Service Providers to monitor the quality of the connections on a MAN or WAN. Service providers can monitor specific events, take actions on events, and if necessary, put specific interfaces into loopback mode for troubleshooting. Ethernet OAM features can be configured to monitor either side or both sides of a link.

For more information on Ethernet OAM protocols, refer to the *Configuring Ethernet Interfaces* chapter of the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

Layer 2 VPN on Ethernet Interfaces

Layer 2 Virtual Private Network (L2VPN) connections emulate the behavior of a LAN across an IP or MPLS-enabled IP network, allowing Ethernet devices to communicate with each other as if they were connected to a common LAN segment.

The L2VPN feature enables service providers (SPs) to provide L2 services to geographically disparate customer sites. Typically, an SP uses an access network to connect the customer to the core network. This access network may use a mixture of L2 technologies, such as Ethernet and Frame Relay. The connection between the customer site and the nearby SP edge router is known as an attachment circuit (AC). Traffic from the customer travels over this link to the edge of the SP core network. The traffic then tunnels through a pseudowire over the SP core network to another edge router. The edge router sends the traffic down another AC to the customer's remote site.

The L2VPN feature enables the connection between different types of L2 attachment circuits and pseudowires, allowing users to implement different types of end-to-end services.

Cisco IOS XR software supports a point-to-point end-to-end service, where two Ethernet circuits are connected together. An L2VPN Ethernet port can operate in one of two modes:

- Port Mode—In this mode, all packets reaching the port are sent over the pseudowire, regardless of any VLAN tags that are present on the packets. In VLAN mode, the configuration is performed under the l2transport configuration mode.
- VLAN Mode—Each VLAN on a CE (customer edge) or access network to PE (provider edge) link can be configured as a separate L2VPN connection (using either VC type 4 or VC type 5). To configure L2VPN on VLANs, see *The Carrier Ethernet Model* chapter in this manual. In VLAN mode, the configuration is performed under the individual subinterface.

Switching can take place in three ways:

- AC-to-PW—Traffic reaching the PE is tunneled over a PW (pseudowire) (and conversely, traffic arriving over the PW is sent out over the AC). This is the most common scenario.
- Local switching—Traffic arriving on one AC is immediately sent out of another AC without passing through a pseudowire.
- PW stitching—Traffic arriving on a PW is not sent to an AC, but is sent back into the core over another PW.

Keep these in mind when configuring L2VPN on an Ethernet interface:

- L2VPN links support QoS (Quality of Service) and MTU (Maximum Transmission Unit) configuration.
- If your network requires that packets are transported transparently, you may need to modify the packet's destination MAC (Media Access Control) address at the edge of the Service Provider (SP) network. This prevents the packet from being consumed by the devices in the SP network.

Use the **show interfaces** command to display AC and pseudowire information.

Gigabit Ethernet Protocol Standards Overview

The Gigabit Ethernet interfaces support these protocol standards:

- IEEE 802.3 Physical Ethernet Infrastructure
- IEEE 802.3ab 1000BASE-T Gigabit Ethernet
- IEEE 802.3z 1000 Mbps Gigabit Ethernet
- IEEE 802.3ae 10 Gbps Ethernet

These standards are further described in the sections that follow.

IEEE 802.3 Physical Ethernet Infrastructure

The IEEE 802.3 protocol standards define the physical layer and MAC sublayer of the data link layer of wired Ethernet. IEEE 802.3 uses Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access at a variety of speeds over a variety of physical media. The IEEE 802.3 standard covers 10 Mbps Ethernet. Extensions to the IEEE 802.3 standard specify implementations for Gigabit Ethernet, 10-Gigabit Ethernet, and Fast Ethernet.

IEEE 802.3ab 1000BASE-T Gigabit Ethernet

The IEEE 802.3ab protocol standards, or Gigabit Ethernet over copper (also known as 1000BaseT) is an extension of the existing Fast Ethernet standard. It specifies Gigabit Ethernet operation over the Category 5e/6 cabling systems already installed, making it a highly cost-effective solution. As a result, most copper-based environments that run Fast Ethernet can also run Gigabit Ethernet over the existing network infrastructure to dramatically boost network performance for demanding applications.

IEEE 802.3z 1000 Mbps Gigabit Ethernet

Gigabit Ethernet builds on top of the Ethernet protocol, but increases speed tenfold over Fast Ethernet to 1000 Mbps, or 1 Gbps. Gigabit Ethernet allows Ethernet to scale from 10 or 100 Mbps at the desktop to 100 Mbps up to 1000 Mbps in the data center. Gigabit Ethernet conforms to the IEEE 802.3z protocol standard.

By leveraging the current Ethernet standard and the installed base of Ethernet and Fast Ethernet switches and routers, network managers do not need to retrain and relearn a new technology in order to provide support for Gigabit Ethernet.

IEEE 802.3ae 10 Gbps Ethernet

Under the International Standards Organization's Open Systems Interconnection (OSI) model, Ethernet is fundamentally a L2 protocol. 10-Gigabit Ethernet uses the IEEE 802.3 Ethernet MAC protocol, the IEEE 802.3 Ethernet frame format, and the minimum and maximum IEEE 802.3 frame size. 10 Gbps Ethernet conforms to the IEEE 802.3ae protocol standards.

Just as 1000BASE-X and 1000BASE-T (Gigabit Ethernet) remained true to the Ethernet model, 10-Gigabit Ethernet continues the natural evolution of Ethernet in speed and distance. Because it is a full-duplex only and fiber-only technology, it does not need the carrier-sensing multiple-access with the CSMA/CD protocol that defines slower, half-duplex Ethernet technologies. In every other respect, 10-Gigabit Ethernet remains true to the original Ethernet model.

General Ethernet Standards

- Ethernet II framing also known as DIX.
- IEEE 802.3 framing also includes LLC and LLC/SNAP protocol frame formats.
- IEEE 802.1d MAC Bridges and Spanning Tree—This standard specifies the MAC learning and MAC aging in a bridging environment. It also defines the original spanning tree protocol. Also MSTP is defined in IEEE 802.1s and IEEE 802.1q.
- IEEE 802.1q VLAN tagging—This standard defines VLAN tagging, and also the traditional VLAN trunking between switches. Technically, it also defines QinQ tagging, and MSTP. The Cisco ASR 9000 Series Routers do NOT support ISL.
- IEEE 802.1ad Provider Bridges—This standard is a subset of 802.1q and is often referred to as 802.1ad. The Cisco ASR 9000 Series Routers do not adhere to the entire standard, but large portions of the standard's functionality are supported.

MAC Address

A MAC address is a unique 6-byte address that identifies the interface at L2.

Ethernet MTU

The Ethernet Maximum Transmission Unit (MTU) is the size of the largest frame, minus the 4-byte Frame Check Sequence (FCS), that can be transmitted on the Ethernet network. Every physical network along the destination of a packet can have a different MTU.

Cisco IOS XR software supports two types of frame forwarding processes:

- Fragmentation for IPV4 packets—In this process, IPv4 packets are fragmented as necessary to fit within the MTU of the next-hop physical network.

Note IPv6 does not support fragmentation.

- MTU discovery process determines largest packet size—This process is available for all IPV6 devices, and for originating IPv4 devices. In this process, the originating IP device determines the size of the largest IPv6 or IPV4 packet that can be sent without being fragmented. The largest packet is equal to the smallest MTU of any network between the IP source and the IP destination devices. If a packet is larger than the smallest MTU of all the networks in its path, that packet will be fragmented as necessary. This process ensures that the originating device does not send an IP packet that is too large.

Jumbo frame support is automatically enable for frames that exceed the standard frame size. The default value is 1514 for standard frames and 1518 for 802.1Q tagged frames. These numbers exclude the 4-byte FCS.

Flow Control on Ethernet Interfaces

The flow control used on 10-Gigabit Ethernet interfaces consists of periodically sending flow control pause frames. It is fundamentally different from the usual full- and half-duplex flow control used on standard management interfaces. On the Cisco ASR 9000 Series Routers both ingress and egress flow control are off by default.

VRRP

The Virtual Router Redundancy Protocol (VRRP) eliminates the single point of failure inherent in the static default routed environment. VRRP specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VPN concentrators on a LAN. The VRRP VPN concentrator controlling the IP addresses associated with a virtual router is called the primary, and forwards packets sent to those IP addresses. When the primary becomes unavailable, a backup VPN concentrator takes the place of the primary.

For more information on VRRP, see the *Implementing VRRP chapter of Cisco ASR 9000 Series Routers IP Addresses and Services Configuration Guide*.

HSRP

Hot Standby Routing Protocol (HSRP) is a proprietary protocol from Cisco. HSRP is a routing protocol that provides backup to a router in the event of failure. Several routers are connected to the same segment of an Ethernet, FDDI, or token-ring network and work together to present the appearance of a single virtual router on the LAN. The routers share the same IP and MAC addresses and therefore, in the event of failure of one router, the hosts on the LAN are able to continue forwarding packets to a consistent IP and MAC address. The transfer of routing responsibilities from one device to another is transparent to the user.

HSRP is designed to support non disruptive failover of IP traffic in certain circumstances and to allow hosts to appear to use a single router and to maintain connectivity even if the actual first hop router they are using fails. In other words, HSRP protects against the failure of the first hop router when the source host cannot learn the IP address of the first hop router dynamically. Multiple routers participate in HSRP and in concert create the illusion of a single virtual router. HSRP ensures that one and only one of the routers is forwarding packets on behalf of the virtual router. End hosts forward their packets to the virtual router.

The router forwarding packets is known as the *active router*. A standby router is selected to replace the active router should it fail. HSRP provides a mechanism for determining active and standby routers, using the IP addresses on the participating routers. If an active router fails, a standby router can take over without a major interruption in the host's connectivity.

HSRP runs on top of User Datagram Protocol (UDP), and uses port number 1985. Routers use their actual IP address as the source address for protocol packets, not the virtual IP address, so that the HSRP routers can identify each other.

For more information on HSRP, see the *Implementing HSRP chapter of Cisco ASR 9000 Series Routers IP Addresses and Services Configuration Guide*.

Link Autonegotiation on Ethernet Interfaces

Link autonegotiation ensures that devices that share a link segment are automatically configured with the highest performance mode of interoperation. Use the **negotiation auto** command in interface configuration mode to enable link autonegotiation on an Ethernet interface. On line card Ethernet interfaces, link autonegotiation is disabled by default.

Note The **negotiation auto** command is available on Gigabit Ethernet interfaces only.

What is an Ethernet Flow Point?

An Ethernet Flow Point (EFP) is a Layer 2 logical subinterface used to classify traffic under a physical or a bundle interface.

A physical interface can be a Gigabit Ethernet 0/0/0/1 or a 10 Gigabit Ethernet 0/0/0/0 interface and has ports on the line card. A bundle interface is a virtual interface, created by grouping physical interfaces together.

For example, physical interfaces such as Gigabit Ethernet 0/0/0/1 and 10 Gigabit Ethernet 0/0/0/0 can be configured as members of a bundle interface.

Grouping physical interfaces together can:

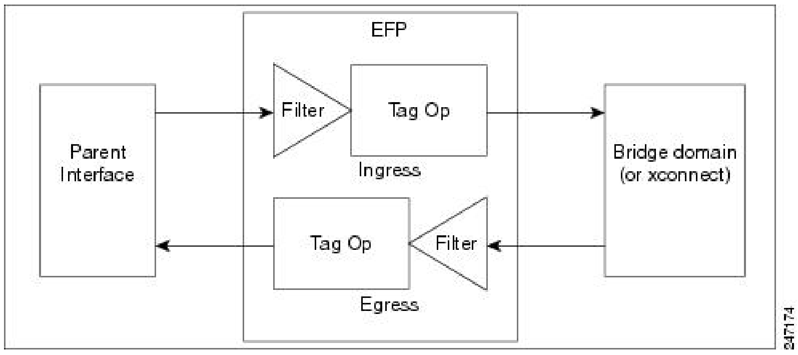
- Reduce the routing entries
- Increase the bandwidth of the bundle interface
- Balance the traffic on the bundle members

EFP has the following characteristics:

- An EFP represents a logical demarcation point of an Ethernet Virtual Connection (EVC) on an interface. For an EVC associating two or more UNIs, there is a flow point on each interface of every device, through which that EVC passes.
- An EFP can be regarded as an instantiation of a particular service. An EFP is defined by a set of filters. These filters are applied to all the ingress traffic to classify the frames that belong to a particular EFP. An EFP *filter* is a set of entries, where each entry looks similar to the start of a packet (ignoring source/destination MAC address). Each entry usually contains 0, 1 or 2 VLAN tags. A packet that starts with the same tags as an entry in the filter is said to match the filter; if the start of the packet does not correspond to any entry in the filter then the packet does not match the filter.
- An EFP serves four purposes:
    - Identifies all frames that belong to a particular flow on a given interface
    - Performs ingress and egress Ethernet header manipulations
    - Adds features to the identified frames
    - Optionally define how to forward those frames in the data path

You can perform a variety of operations on the traffic flows when a router is configured with EFPs on various interfaces. Also, you can bridge or tunnel the traffic by many ways from one or more of the router's ingress EFPs to one or more egress EFPs. This traffic is a mixture of VLAN IDs, single or double (QinQ) encapsulation, and ethertypes.

Figure 3: EFP Model



An EFP subinterface is configured to specify which traffic on ingress is vectored to that EFP. This is done by specifying a VLAN, range of VLANs, or QinQ tagging to match against on ingress. All traffic on ingress is compared to each EFP's matching criterion, and processed by that EFP if a match occurs. The processing performed by an EFP can change VLAN IDs, add or remove VLAN tags, and change ethertypes.

Improving the Scalability of EFPs on Bundle Interfaces

You can improve the scalability of EFPs on bundle interfaces in two ways:

- Increase the number of EFPs per chassis from 32000 to 64000.
- Increase the number of EFPs per line card, on a single node point, to the same scale as the physical interface scaling.

The following example illustrates how to improve the scalability of EFPs per line card:

Consider a B module line card type with a bundle interface scaling of 4000 and a physical interface scaling of 16000. The scalability of EFPs on the B module is improved by adding three additional bundles of 4000 EFPs per bundle.

The number of EFPs per line card is now scaled to 16000 or 4 bundles of 4000 EFPs each.

Note The maximum number of EFPs that can be added to a bundle interface is 4000.

EFP CLI Overview

Cisco IOS XR implements a structured CLI for EFP and EVC configuration. The following commands are typically used to configure an EFP:

- **l2transport** command - This command identifies a subinterface (or a physical port or bundle-port parent interface) as an EFP.
- **encapsulation** command - This command is used to specify matching criteria.
- **rewrite** command - This command is used to specify the VLAN tag rewrite criteria.

Egress EFP Filtering

The Egress EFP Filtering feature implements a means of filtering EFP egress traffic, ensuring that all the given EFP's egress traffic complies with the ingress matching criterion.

An ingress EFP is similar to an egress EFP. The router is configured to send traffic on the EFP, that matches that EFP's ingress matching criterion. It is possible to configure a router so that this does not occur, and there is no safeguard to prevent such mismatching egress EFP traffic from exiting the router.

The Cisco ASR 9000 Series Routers allow for different VLANs on different ports within the same bridge domain. This allows a bridge to forward a packet out of a port not configured for the VLAN tag on the packet. Egress EFP filtering checks this and drops invalid packets at the egress port.

Identifying Frames of an EFP

The EFP identifies frames belonging to a particular flow on a given port, independent of their Ethernet encapsulation. An EFP can flexibly map frames into a flow or EFP based on the fields in the frame header.

The frames can be matched to an EFP using:

- VLAN tag or tags
- MAC address (source address, destination address, or both)
- 802.1p CoS bits
- Logical conjunction of two or more of the above: VLAN, MAC, and CoS
- Default match (that is, any other traffic that has not matched a more specific EFP)
- Protocol ethertype

The frames cannot be matched to an EFP through use of any of these:

- Any information outside the outermost Ethernet frame header and its associated tags such as
    - IPv4, IPv6, or MPLS tag header data
    - C-DMAC, C-SMAC, or C-VLAN
- Logical disjunction of the valid frame matches above: VLAN, MAC, and CoS

The specific match criteria are covered in more detail in these sections.

VLAN Tag Matching

Below table describes the different encapsulation types and the EFP identifier corresponding to each.

Encapsulation Type	EFP Identifier
Untagged	Static configuration on the ingress physical interface or a subinterface that uses the **untagged** keyword in the **encapsulation** command. There can be only one untagged subinterface. If an untagged subinterface has been created, traffic goes to this interface instead of the main interface.

Encapsulation Type	EFP Identifier
Priority-tagged Ethernet frames	A priority-tagged frame is defined as having a single 802.1Q VLAN header, with a VLAN id of zero.
Native VLAN	Cisco ASR 9000 Series Routers do not support native VLAN. Use this command: **encapsulation dot1q** <*vlan-id*>, **untagged**
Single tagged frames	802.1Q customer-tagged Ethernet frames
Double tagged frames	802.1Q (ethertype 0x8100) double tagged frames 802.1ad double tagged frames Legacy 0x9100 and 0x9200 double tagged frames
Default tagging	An EFP which has a maximum-match wildcard. The effect is to receive any traffic that does not match any other EFP on the same physical interface.

You can use wildcards as well as VLAN ranges while defining frames that map to a given EFP. EFPs can distinguish flows based on a single VLAN tag, a range of VLAN tags, a stack of VLAN tags or a combination of both (VLAN stack with wildcards). It provides the EFP model, a flexibility of being encapsulation agnostic, and allows it to be extensible as new tagging or tunneling schemes are added.

MAC Address Matching

The source MAC address, the destination MAC address, or both can be matched. In all cases, the MAC address requires an exact match. A wildcard match or partial match is not adequate.

802.1p CoS Bits Matching

One or more exact CoS matches are specified. Because CoS is only 3 bits, this limits it to 8 possible choices.

Logical Conjunction

All of the match criteria above can be selectively combined those frames that match all of the separate criteria.

Default Match

A single EFP can be defined that matches all other traffic that has not been matched by a more specific EFP.

Match Precedence and Config Verification

Overlapping EFPs are allowed to be configured, where it is possible to determine an order in which they should be used for matching. But EFPs that conflict with other EFPs or subinterfaces on the parent trunk interface should be blocked at config verification.

An ordering precedence is used for how EFP matches are applied in hardware. The model is for matches that are more specific to be processed before matches that are less specific.

Egress Behavior

The EFP matching criteria can also be used on egress to police the frames that can egress from the EFP, based on the platform support. Frames that do not match the criteria (source/destination MAC match criteria are reversed) are dropped.

Applying Features

After the frames are matched to a particular EFP, any appropriate features can be applied. In this context, “features” means any frame manipulations specified by the configuration as well as things such as QoS and ACLs. The Ethernet infrastructure provides an appropriate interface to allow the feature owners to apply their features to an EFP. Hence, IM interface handles are used to represent EFPs, allowing feature owners to manage their features on EFPs in the same way the features are managed on regular interfaces or subinterfaces.

The only L2 features that can be applied on an EFP that is part of the Ethernet infrastructure are the L2 header encapsulation modifications. The L2 features are described in this section.

Encapsulation Modifications

EFP supports these L2 header encapsulation modifications on both ingress and egress:

- Push 1 or 2 VLAN tags
- Pop 1 or 2 VLAN tags

Note This modification can only pop tags that are matched as part of the EFP.

- Rewrite 1 or 2 VLAN tags:
    - Rewrite outer tag
    - Rewrite outer 2 tags
    - Rewrite outer tag and push an additional tag
    - Remove outer tag and rewrite inner tag

Note For dot1ad encapsulation, priority-tagged frame and rewrite ingress tag cannot coexist.

For each of the VLAN ID manipulations, these can be specified:

- The VLAN tag type, that is, C-VLAN, S-VLAN, or I-TAG. The ethertype of the 802.1Q C-VLAN tag is defined by the dot1q tunneling type command.
- The VLAN ID. 0 can be specified for an outer VLAN tag to generate a priority-tagged frame.

Note For tag rewrites, the CoS bits from the previous tag should be preserved in the same way as the DEI bit for 802.1ad encapsulated frames.

Defining Data-Forwarding Behavior

The EFP can be used to designate the frames belonging to a particular Ethernet flow forwarded in the data path. These forwarding cases are supported for EFPs in Cisco IOS XR software:

- L2 Switched Service (Bridging)—The EFP is mapped to a bridge domain, where frames are switched based on their destination MAC address. This includes multipoint services:
    - Ethernet to Ethernet Bridging
    - Multipoint Layer 2 Services
- L2 Stitched Service (AC to AC xconnect)—This covers point-to-point L2 associations that are statically established and do not require a MAC address lookup.
    - Ethernet to Ethernet Local Switching—The EFP is mapped to an S-VLAN either on the same port or on another port. The S-VLANs can be identical or different.
- Tunneled Service (xconnect)—The EFP is mapped to a Layer 3 tunnel. This covers point-to-point services only:
    - EoMPLS
    - L2TPv3
- L2 Terminated Service (Ethernet access to Layer 3 service)—The EFP is mapped to an IP interface that has a global address or belongs to a VRF (includes both IP and MPLS Layer 3 VPNs).

802.1Q VLAN

A VLAN is a group of devices on one or more LANs that are configured so that they can communicate as if they were attached to the same wire, when in fact they are located on a number of different LAN segments. Because VLANs are based on logical instead of physical connections, it is very flexible for user and host management, bandwidth allocation, and resource optimization.

The IEEE's 802.1Q protocol standard addresses the problem of breaking large networks into smaller parts so broadcast and multicast traffic does not consume more bandwidth than necessary. The standard also helps provide a higher level of security between segments of internal networks.

The 802.1Q specification establishes a standard method for inserting VLAN membership information into Ethernet frames.

Cisco IOS XR software supports VLAN subinterface configuration on Gigabit Ethernet and10-Gigabit Ethernet interfaces.

802.1Q Tagged Frames

The IEEE 802.1Q tag-based VLAN uses an extra tag in the MAC header to identify the VLAN membership of a frame across bridges. This tag is used for VLAN and Quality of Service (QoS) priority identification. The VLANs can be created statically by manual entry or dynamically through Generic Attribute Registration Protocol (GARP) VLAN Registration Protocol (GVRP). The VLAN ID associates a frame with a specific VLAN and provides the information that switches must process the frame across the network. A tagged frame is four bytes longer than an untagged frame and contains two bytes of Tag Protocol Identifier (TPID) residing within the type and length field of the Ethernet frame and two bytes of Tag Control Information (TCI) which starts after the source address field of the Ethernet frame.

Subinterfaces

Subinterfaces are logical interfaces created on a hardware interface. These software-defined interfaces allow for segregation of traffic into separate logical channels on a single hardware interface as well as allowing for better utilization of the available bandwidth on the physical interface.

Subinterfaces are distinguished from one another by adding an extension on the end of the interface name and designation. For instance, the Ethernet subinterface 23 on the physical interface designated TenGigE 0/1/0/0 would be indicated by TenGigE 0/1/0/0.23.

Before a subinterface is allowed to pass traffic it must have a valid tagging protocol encapsulation and VLAN identifier assigned. All Ethernet subinterfaces always default to the 802.1Q VLAN encapsulation. However, the VLAN identifier must be explicitly defined.

Subinterface MTU

The subinterface Maximum Transmission Unit (MTU) is inherited from the physical interface with an additional four bytes allowed for the 802.1Q VLAN tag.

VLAN Subinterfaces on Ethernet Bundles

An Ethernet bundle is a group of one or more Ethernet ports that are aggregated together and treated as a single link. Multiple VLAN subinterfaces can be added to a single Ethernet bundle.

For more information about configuring Ethernet bundles, see the Configuring Link Bundles chapter in this document. The procedure for creating VLAN subinterfaces on an Ethernet bundle is exactly the same as the procedure for creating VLAN subinterfaces on a physical Ethernet interface.

To create a VLAN subinterface on an Ethernet bundle, see the Configuring 802.1Q VLAN Interfaces section later in this chapter.

Layer 2 VPN on VLANs

The Layer 2 Virtual Private Network (L2VPN) feature enables Service Providers (SPs) to provide L2 services to geographically disparate customer sites, as described in the Layer 2 VPN on Ethernet Interfaces section of the The Carrier Ethernet Model, on page 3 chapter.

The configuration model for configuring VLAN Attachment Circuits (ACs) is similar to the model used for configuring basic VLANs, where the user first creates a VLAN subinterface, and then configures that VLAN in subinterface configuration mode. To create an Attachment Circuit, you need to include the **l2transport** keyword in the **interface** command string to specify that the interface is a L2 interface.

VLAN ACs support three modes of L2VPN operation:

- Basic Dot1Q Attachment Circuit—The Attachment Circuit covers all frames that are received and sent with a specific VLAN tag.
- QinQ Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and a specific inner VLAN tag. QinQ is an extension to Dot1Q that uses a stack of two tags.
- Q-in-Any Attachment Circuit—The Attachment Circuit covers all frames received and sent with a specific outer VLAN tag and any inner VLAN tag, as long as that inner VLAN tag is not Layer 3 terminated. Q-in-Any is an extension to QinQ that uses wildcarding to match any second tag.

Note The Q-in-Any mode is a variation of the basic Dot1Q mode. In Q-in-Any mode, the frames have a basic QinQ encapsulation; however, in Q-in-Any mode the inner tag is not relevant, except for the fact that a few specific inner VLAN tags are siphoned for specific services. For example, a tag may be used to provide L3 services for general internet access.

Each VLAN on a CE-to-PE link can be configured as a separate L2VPN connection (using either VC type 4 or VC type 5).

Keep these in mind when configuring L2VPN on a VLAN:

- Cisco IOS XR software supports 4000 Attachment Circuits per line card.
- In a point-to-point connection, the two Attachment Circuits do not have to be of the same type. For example, a port mode Ethernet Attachment Circuit can be connected to a Dot1Q Ethernet Attachment Circuit.
- Pseudowires can run in VLAN mode or in port mode. A pseudowire running in VLAN mode has a single Dot1Q tag, while a pseudowire running in port mode has no tags. Some interworking is required to connect these different types of circuits together. This interworking takes the form of popping, pushing, and rewriting tags. The advantage of L2VPN is that is simplifies the interworking required to connect completely different media types together.
- The Attachment Circuits on either side of an MPLS pseudowire can be different types. In this case, the appropriate conversion is carried out at one or both ends of the Attachment Circuit to pseudowire connection.

Use the **show interfaces** command to display Attachment Circuit and pseudowire information.

Note For more information on the **show interfaces** command, refer to the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

How to Configure Layer 2 Features on Ethernet Interfaces

Note For more information on configuring interfaces, refer to the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

Default Configuration Values for Gigabit Ethernet and 10-Gigabit Ethernet

The below table describes the default interface configuration parameters that are present when an interface is enabled on a Gigabit Ethernet or 10-Gigabit Ethernet modular services card and its associated PLIM.

Note You must use the **shutdown** command to bring an interface administratively down. The interface default is **no shutdown**. When a modular services card is first inserted into the router, if there is no established preconfiguration for it, the configuration manager adds a shutdown item to its configuration. This shutdown can be removed only be entering the **no shutdown** command.

Table 2: Gigabit Ethernet and 10-Gigabit Ethernet Modular Services Card Default Configuration Values

Parameter	Configuration File Entry	Default Value	Restrictions
Flow control	**flow-control**	egress on ingress off	none
MTU	**mtu**	1514 bytes for normal frames 1518 bytes for 802.1Q tagged frames 1522 bytes for QinQ frames	none
MAC address	**mac address**	Hardware burned-in address (BIA[2])	L3 only
L2 port	**l2transport**	off/L3	L2 subinterfaces must have L3 main parent interface
Egress filtering	**Ethernet egress-filter**	off	none
Link negotiation	**negotiation**	off	physical main interfaces only
Tunneling Ethertype	**tunneling ethertype**	0X8100	configured on main interface only; applied to subinterfaces only
VLAN tag matching	**encapsulation**	all frames for main interface; only ones specified for subinterfaces	encapsulation command only subinterfaces

1. The restrictions are applicable to L2 main interface, L2 subinterface, L3 main interface, interflex L2 interface etc.
2. burned-in address

Configuring Ethernet Interfaces

For more information on configuring Ethernet interfaces, see the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

Configuring a 10-Gigabit Ethernet Interface

Perform this task to configure an Ethernet interface:

SUMMARY STEPS

1. **configure interface TenGigE** [*instance*]
2. **l2transport**
3. **mtu** *bytes*
4. **no shutdown**
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure interface TenGigE** [*instance*]

Example:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router# interface TenGigE 0/0/0/1
```

Enters interface configuration mode for a 10-Gigabit Ethernet interface.

Step 2 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# l2transport
```

Enables Layer 2 transport mode on a port and enter Layer 2 transport configuration mode.

Step 3 **mtu** *bytes*

Example:

```
RP/0/RSP0/CPU0:router(config-if-l2)# mtu 1448
```

Adjusts the maximum packet size or Maximum Transmission Unit (MTU) size for the bridge domain.

- Use the bytes argument to specify the MTU size, in bytes. The range is from 64 to 65535.

Step 4 **no shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-if-l2)#  no shutdown
```

Removes the shutdown configuration, which forces an interface administratively down.

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

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

end - Prompts user to take one of these actions:

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

Configuring a Gigabit Ethernet Interface

Perform this task to configure a basic Gigabit Ethernet or 10-Gigabit Ethernet interface:

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 address** *ip-address mask*
4. **flow-control** { **bidirectional** | **egress** | **ingress** }
5. **mtu** *bytes*
6. **mac-address** *value1.value2.value3*
7. **negotiation auto** (on Gigabit Ethernet interfaces only)
8. **no shutdown**
9. Use the **commit** or **end** command.
10. **show interfaces** [**GigabitEthernet** | **TenGigE**] *instance*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet 0/1/0/0
```

Enters interface configuration mode and specifies the Ethernet interface name and notation *rack/slot/module/port*.

Step 3 **ipv4 address** *ip-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 172.18.189.38 255.255.255.224
```

Assigns an IP address and subnet mask to the interface.

- Replace *ip-address* with the primary IPv4 address for the interface.
- Replace *mask* with the mask for the associated IP subnet. The network mask can be specified in either of two ways:
  - The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means that the corresponding address bit belongs to the network address.
  - The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address.

Step 4 **flow-control** { **bidirectional** | **egress** | **ingress** }

Example:

```
RP/0/RSP0/CPU0:router(config-if)# flow control ingress
```

(Optional) Enables the sending and processing of flow control pause frames.

- **egress**—Enables the sending of flow control pause frames in egress.
- **ingress**—Enables the processing of received pause frames on ingress.
- **bidirectional**—Enables the sending of flow control pause frames in egress and the processing of received pause frames on ingress.

Step 5 **mtu** *bytes*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# mtu 1448
```

(Optional) Sets the MTU value for the interface.

- The default is 1514 bytes for normal frames and 1518 bytes for 802.1Q tagged frames.
- The range for Gigabit Ethernet and 10-Gigabit Ethernet mtu values is 64 bytes to 65535 bytes.

Step 6 **mac-address** *value1.value2.value3*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# mac address 0001.2468.ABCD
```

(Optional) Sets the MAC layer address of the Management Ethernet interface.

- The values are the high, middle, and low 2 bytes, respectively, of the MAC address in hexadecimal. The range of each 2-byte value is 0 to ffff.

Step 7 **negotiation auto** (on Gigabit Ethernet interfaces only)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# negotiation auto
```

(Optional) Enables autonegotiation on a Gigabit Ethernet interface.

- Autonegotiation must be explicitly enabled on both ends of the connection, or speed and duplex settings must be configured manually on both ends of the connection.
- If autonegotiation is enabled, any manually configured speed or duplex settings take precedence.

Note The **negotiation auto** command is available on Gigabit Ethernet interfaces only.

Step 8 **no shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# no shutdown
```

Removes the shutdown configuration, which forces an interface administratively down.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Step 10 **show interfaces** [**GigabitEthernet** | **TenGigE**] *instance*

Example:

```
RP/0/RSP0/CPU0:router #show interfaces TenGigE 0/3/0/0
```

(Optional) Displays statistics for interfaces on the router.

What to Do Next

- To configure an 802.1Q VLAN subinterface on the Ethernet interface, see the "The Carrier Ethernet Model" chapter later in this manual.
- To configure an AC on the Ethernet port for L2VPN implementation, see the "Configuring an Attachment Circuit on an Ethernet Port" section later in this chapter.

Configuring an Attachment Circuit on an Ethernet Port

Use this procedure to configure an attachment circuit on a Gigabit Ethernet or 10-Gigabit Ethernet port. For more information on configuring an attachment circuit, refer to the *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

Note The steps in this procedure configure the L2VPN Ethernet port to operate in EFP mode.

SUMMARY STEPS

1. **configure**
2. **interface [GigabitEthernet | TenGigE]** *instance.subinterface* **l2transport**
3. **encapsulation dot1q** *vlan-id*
4. **interface [GigabitEthernet | TenGigE]** *instance.subinterface* **l2transport**
5. **encapsulation dot1q** *vlan-id*
6. **l2vpn**
7. **bridge group** *bridge-group-name*
8. **bridge-domain** *domain-name*
9. **interface [GigabitEthernet | TenGigE]** *instance.subinterface*
10. **interface [GigabitEthernet | TenGigE]** *instance.subinterface*
11. Use the **commit** or **end** command.
12. **show run interface [GigabitEthernet | TenGigE]** *instance.subinterface*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface [GigabitEthernet | TenGigE]** *instance.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/5/0/0.20
l2transport
```

Enters subinterface configuration mode and specifies the interface type, location, and subinterface number.

- Replace the instance argument with one of these instances:
  - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is rack/slot/module/port, and a slash between values is required as part of the notation.
  - Ethernet bundle instance. Range is from 1 through 65535.
- Replace the subinterface argument with the subinterface value. Range is from 0 through 4095.
- Naming notation is instance.subinterface, and a period between arguments is required as part of the notation.

Step 3 **encapsulation dot1q** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-subif)#encapsulation dot1q 50
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 4 **interface [GigabitEthernet | TenGigE]** *instance.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)#  interface GigabitEthernet0/5/0/0.20
l2transport
```

Enters subinterface configuration mode and specifies the interface type, location, and subinterface number.

- Replace the instance argument with one of these instances:
  - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is rack/slot/module/port, and a slash between values is required as part of the notation.
  - Ethernet bundle instance. Range is from 1 through 65535.
- Replace the subinterface argument with the subinterface value. Range is from 0 through 4095.
- Naming notation is instance.subinterface, and a period between arguments is required as part of the notation.

Step 5 **encapsulation dot1q** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 50
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 6 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
```

Enters L2VPN configuration mode.

Step 7 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group ce-doc-examples
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 8 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-
domain ac-example
```

Enters configuration mode for the named bridge domain. This creates a new bridge domain modifies the existing bridge domain if it already exists.

Step 9 **interface [GigabitEthernet | TenGigE]** *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/5/0/0.20
```

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. The interface EFP now becomes an attachment circuit on this bridge domain.

Step 10 **interface [GigabitEthernet | TenGigE]** *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# interface GigabitEthernet0/5/0/1.15
```

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. The interface EFP now becomes an attachment circuit on this bridge domain.

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

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

end - Prompts user to take one of these actions:

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

Step 12 **show run interface [GigabitEthernet | TenGigE]** *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:router# show run interface GigabitEthernet0/5/0/1.15
```

(Optional) Displays statistics for the subinterface on the router.

Configuring Egress EFP Filtering

This section describes the procedures for configuring the egress EFP filtering feature on the Cisco ASR 9000 Series Routers.

Egress EFP filtering is a L2 subinterface specific feature that controls how strictly subinterface encapsulation filtering is performed in the egress direction. According to the EFP behavior and model, all packets transmitted out of a subinterface should match the subinterface encapsulation or rewrite criteria if the same packet is to be received on the subinterface (with the source and destination MAC addresses swapped).

Egress EFP filtering has two stages; first stage is without rewrite command, and the second stage is with rewrite command.

In the first stage filtering, the packet is checked against the encapsulation to ensure the match, the same way it is checked on ingress to determine that the packet is forwarded to that EFP.

In the second stage filtering, the packet is checked before the egress rewrite occurs to ensure that the packet in its egress pre-rewrite state is correct. This means that the egress packet's VLAN encapsulation should be same as a hypothetical ingress packet after the ingress rewrite occurs.

In case of an interface configured with both a rewrite and egress EFP filtering, where egress traffic is getting dropped unexpectedly due to egress EFP filtering, the user must first ascertain which stage the drops occur.

Note Output drops counter displays the drops occurred due to egress EFP filtering in the “show interface” display for that interface. Output drops counter is a summation of drops from multiple causes and not necessarily due to egress EFP filtering.

By using the **ethernet egress-filter** command, you can configure egress EFP filtering in either global or L2 subinterface mode:

- **ethernet egress-filter strict** configures Egress EFP Filtering inGlobal Configuration mode.
- **ethernet egress-filter {strict | disabled}** configures Egress EFP Filtering in L2 subinterface mode.

SUMMARY STEPS

1. **configure**
2. **ethernet egress-filter strict**
3. **interface {GigabitEthernet | TenGigE | FastEthernet | Bundle-Ether}** *instance.subinterface*
4. **ethernet egress-filter {strict | disabled}**
5. **exit**

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:PE44_ASR-9010# config Thu Jun 4 07:50:02.660 PST
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#
```

Enters Global Configuration mode.

Step 2 **ethernet egress-filter strict**

Example:

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#  ethernet egress-filter strict
```

Enables strict egress filtering on all subinterfaces on the device by default.

Step 3 **interface {GigabitEthernet | TenGigE | FastEthernet | Bundle-Ether}** *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config)#  interface GigabitEthernet 0/1/0/1.1
 RP/0/RSP0/CPU0:PE44_ASR-9010(config-subif)#
```

Creates an L2 subinterface.

Step 4 **ethernet egress-filter {strict | disabled}**

Example:

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config-subif)# ethernet egress-filter strict
```

Allows egress filtering to be explicitly enabled or disabled on any L2 subinterface. It can also be used to override global settings.

Step 5 **exit**

Example:

```
RP/0/RSP0/CPU0:PE44_ASR-9010(config-subif)# exit
RP/0/RSP0/CPU0:PE44_ASR-9010(config)# exit
```

Exit from the configuration mode.

Configuring 802.1Q VLAN Interfaces

Restrictions for Configuring 802.1Q VLAN Interfaces

To use the sub-interface configurations **encapsulation dot1ad** (or **encapsulation dot1q**) and **encapsulation dot1ad second-dot1q any** (or **encapsulation dot1q second-dot1q any**) together on the same physical interface, use the **exact** keyword as shown below. Else, it results in traffic loss.

```
Router(config)# interface hundredGigE 0/0/0/0.0
Router(config-subif)# encapsulation dot1ad 200 exact

Router(config)# interface hundredGigE 0/0/0/0.1
Router(config-subif)# encapsulation dot1ad 200 dot1q any
Router(config-subif)# commit
```

Configuring 802.1Q VLAN Subinterfaces

This task explains how to configure 802.1Q VLAN subinterfaces. To remove these subinterfaces, see the "Removing an 802.1Q VLAN Subinterface" section of this chapter.

SUMMARY STEPS

1. **configure**
2. **interface** {**GigabitEthernet** | **TenGigE** | **Bundle-Ether**} *instance.subinterface*
3. **l2transport**
4. **encapsulation dot1q** *vlan-id*
5. Use the **commit** or **end** command.

6. **show ethernet trunk bundle-ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface** {**GigabitEthernet** | **TenGigE** | **Bundle-Ether**} *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/2/0/4.10
```

Enters subinterface configuration mode and specifies the interface type, location, and subinterface number.

- Replace the *instance* argument with one of these instances:
  - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack/slot/module/port*, and a slash between values is required as part of the notation.
  - Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.
- Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.

Step 3 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# l2transport
```

Enables Layer 2 transport mode on a port and enter Layer 2 transport configuration mode.

Step 4 **encapsulation dot1q** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-subif-l2)# encapsulation dot1q 100
```

Assigns a VLAN Attachment Circuit to the subinterface.

- Replace the vlan-id argument with a subinterface identifier. Range is from 1 to 4094 inclusive (0 and 4095 are reserved). To configure a basic Dot1Q Attachment Circuit, use this syntax:

  encapsulation dot1q `vlan-id`

- To configure a QinQ Attachment Circuit, use this syntax:

  encapsulation dot1q `vlan-id` second-dot1q `vlan-id`

Note Following are the varieties of **encapsulation** commands:

- encapsulation dot1q 100
- encapsulation dot1q 100 second-dot1q 101
- encapsulation dot1ad 200 dot1q 201

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

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

end - Prompts user to take one of these actions:

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

Step 6 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RSP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Configuring Native VLAN

This task explains how to configure a native VLAN on an interface.

SUMMARY STEPS

1. **configure**
2. **interface** [**GigabitEthernet** | **TenGigE** | **Bundle-Ether**] *instance.subinterface* **l2transport**
3. **encapsulation** [**dot1q** *vlan-id*, **untagged**]
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **interface** [**GigabitEthernet** | **TenGigE** | **Bundle-Ether**] *instance.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet 0/2/0/4.2 l2transport
```

Enters subinterface configuration mode and specifies the interface type, location, and subinterface number.

- Replace the instance argument with one of these instances:
    - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack/slot/module/port,* and a slash between values is required as part of the notation.
    - Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.
- Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.

Note You must include the l2transport keyword in the command string; otherwise, the configuration creates a Layer 3 subinterface rather than an Attachment Circuit.

Step 3 **encapsulation** [**dot1q** *vlan-id*, **untagged**]

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 400
```

Defines the Native VLAN, associated with an 802.1Q trunk interface.

- The *vlan-id* argument is the ID of the subinterface.
- Range is from 1 through 4094 inclusive (0 and 4095 are reserved).

It is possible to receive both dot1q 400 and untagged frames by issuing the **encapsulation** command with the **untagged** keyword.

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

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

end - Prompts user to take one of these actions:

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

Removing an 802.1Q VLAN Subinterface

This task explains how to remove 802.1Q VLAN subinterfaces that have been previously configured using the “Configuring 802.1Q VLAN Subinterfaces" task in this chapter.

SUMMARY STEPS

1. **configure**
2. **no interface {GigabitEthernet | TenGigE | Bundle-Ether}** *instance.subinterface*
3. Repeat Step 2 to remove other VLAN subinterfaces.

4. Use the **commit** or **end** command.
5. **show ethernet trunk bundle-ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **no interface {GigabitEthernet | TenGigE | Bundle-Ether}** *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config)# no interface TenGigE 0/2/0/4.10
```

Removes the subinterface, which also automatically deletes all the configuration applied to the subinterface.

- Replace the *instance* argument with one of these instances:
  - Physical Ethernet interface instance, or with an Ethernet bundle instance. Naming notation is *rack/slot/module/port,* and a slash between values is required as part of the notation.
  - Ethernet bundle instance. Range is from 1 through 65535.
- Replace the *subinterface* argument with the subinterface value. Range is from 0 through 4095.

Naming notation is *instance.subinterface*, and a period between arguments is required as part of the notation.

Step 3 Repeat Step 2 to remove other VLAN subinterfaces.

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

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

end - Prompts user to take one of these actions:

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

Step 5 **show ethernet trunk bundle-ether** *instance*

Example:

```
RP/0/RSP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Configuration Examples

Configuring an Ethernet Interface: Example

This example shows how to configure an interface for a 10-Gigabit Ethernet modular services card:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/0/0/1
RP/0/RSP0/CPU0:router(config-if)# l2transport
RP/0/RSP0/CPU0:router(config-if)# mtu 1448
RP/0/RSP0/CPU0:router(config-if)# no shutdown
RP/0/RSP0/CPU0:router(config-if)# end
Uncommitted changes found, commit them? [yes]: yes

RP/0/RSP0/CPU0:router# show interfaces TenGigE 0/0/0/1

TenGigE0/0/0/1 is down, line protocol is down
  Hardware is TenGigE, address is 0001.2468.abcd (bia 0001.81a1.6b23)
  Internet address is 172.18.189.38/27
  MTU 1448 bytes, BW 10000000 Kbit
     reliability 0/255, txload Unknown, rxload Unknown
  Encapsulation ARPA,
  Full-duplex, 10000Mb/s, LR
  output flow control is on, input flow control is on
  loopback not set
  ARP type ARPA, ARP timeout 01:00:00
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
              0 runts, 0 giants, 0 throttles, 0 parity
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
     0 output errors, 0 underruns, 0 applique, 0 resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
```

Configuring a L2VPN AC: Example

This example indicates how to configure a L2VPN AC on an Ethernet interface:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/5/0/0.2 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# ethernet egress-filter strict
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# clear

RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)# interface gigabitethernet 0/5/0/0.2 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# ethernet egress-filter strict
RP/0/RSP0/CPU0:router(config-subif)# interface gigabitethernet 0/5/0/1.100 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# ethernet egress-filter strict
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
```

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group example
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain mybridge
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface gigabitethernet 0/5/0/0.2
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# interface gigabitethernet 0/5/0/1.100
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# exit
RP/0/RSP0/CPU0:router(config-l2vpn)# exit
RP/0/RSP0/CPU0:router(config)# show

Building configuration...
!! IOS XR Configuration 0.0.0
interface GigabitEthernet0/5/0/0.2 l2transport
 encapsulation dot1q 100
 ethernet egress-filter strict
!
interface GigabitEthernet0/5/0/1.100 l2transport
 encapsulation dot1q 100
 ethernet egress-filter strict
!
l2vpn
 bridge group example
  bridge-domain mybridge
   interface GigabitEthernet0/5/0/0.2
   !
   interface GigabitEthernet0/5/0/1.100
   !
  !
 !
end
```

Configuring VPWS with Link Bundles: Example

Physical Interfaces (Port mode)

```
interface Bundle-Ether12
 l2transport
!
interface GigabitEthernet0/1/0/10
 negotiation auto
 l2transport
!
interface GigabitEthernet0/1/0/20
 bundle id 12 mode on
 negotiation auto
!
interface GigabitEthernet0/1/0/21
 bundle id 12 mode on
 negotiation auto
!
 !
l2vpn
 xconnect group test
  p2p test
   interface Bundle-Ether12
   !
   interface GigabitEthernet0/1/0/10
   !
  !
 !
!
```

Sub Interfaces (EFP mode)

```
interface Bundle-Ether12
!
interface Bundle-Ether12.1 l2transport
  encapsulation dot1q 12
 !
!
interface GigabitEthernet0/1/0/10
 negotiation auto
!
interface GigabitEthernet0/1/0/10.1  l2transport
  encapsulation dot1q 12
 !

!
interface GigabitEthernet0/1/0/20
 bundle id 12 mode on
 negotiation auto
!
interface GigabitEthernet0/1/0/21
 bundle id 12 mode on
 negotiation auto
!
 !
l2vpn
xconnect group test
  p2p test
   interface Bundle-Ether12.1
   !
   interface GigabitEthernet0/1/0/10.1
   !
  !
 !
!
```

Configuring Ethernet Bundle with L2 and L3 Services: Example

This example shows how to configure an Ethernet bundle interface with L3 services:

```
configure
interface Bundle-Ether 100
 ipv4 address 12.12.12.2 255.255.255.0

!
```

This example shows how to configure an Ethernet bundle subinterface with L3 services:

```
configure
interface Bundle-Ether 100.1
 ipv4 address 13.13.13.2 255.255.255.0

!
```

This example shows how to configure an Ethernet bundle interface with L2 services:

```
configure
 interface Bundle-Ether 101
 l2transport

!
```

This example shows how to configure an Ethernet bundle interface with L2 services:

```
configure
 interface Bundle-Ether1.1 l2transport
!
```

Configuring VLAN Subinterfaces: Example

This example shows how to create VLAN subinterfaces:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/2/0/4.1 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 20
RP/0/RSP0/CPU0:router(config-subif)# interface TenGigE0/2/0/4.2 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 30
RP/0/RSP0/CPU0:router(config-subif)# interface TenGigE0/2/0/4.3 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 40
RP/0/RSP0/CPU0:router(config-subif)# commit
RP/0/RSP0/CPU0:router(config-subif)# exit
RP/0/RSP0/CPU0:router(config)# exit
```

This example shows how to create two VLAN subinterfaces on an Ethernet bundle at one time:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 1 l2transport
RP/0/RSP0/CPU0:router(config-if-l2)# exit
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether  1.1 l2transport
RP/0/RSP0/CPU0:router(config-subif-l2)#  encapsulation dot1q 10
RP/0/RSP0/CPU0:router(config-subif)# exit
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 1.2 l2transport
RP/0/RSP0/CPU0:router(config-subif-l2)# encapsulation dot1q 20
RP/0/RSP0/CPU0:router(config-subif)# exit
```

This example shows how to create a basic Dot1Q Attachment Circuit:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/2/0/4.1 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 20
RP/0/RSP0/CPU0:router(config-subif)# commit
RP/0/RSP0/CPU0:router(config-subif)# exit
RP/0/RSP0/CPU0:router(config)# exit
```

This example shows how to create a QinQ Attachment Circuit:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/2/0/4.2 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 20 second-dot1q 10
RP/0/RSP0/CPU0:router(config-subif)# commit
RP/0/RSP0/CPU0:router(config-subif)# exit
RP/0/RSP0/CPU0:router(config)# exit
```

This example shows how to create a Q-in-Any Attachment Circuit:

```
RP/0/RSP/CPU0:router# configure
RP/0/RSP/CPU0:router(config)# interface TenGigE 0/2/0/4.3 l2transport
RP/0/RSP/CPU0:router(config-subif)# encapsulation dot1q 30 second-dot1q any
RP/0/RSP/CPU0:router(config-subif)# commit
RP/0/RSP/CPU0:router(config-subif)# exit
RP/0/RSP/CPU0:router(config)# exit
```

Where to Go Next

When you have configured an Ethernet interface, you can configure individual VLAN subinterfaces on that Ethernet interface. For information about configuring VLAN subinterfaces, see the The Carrier Ethernet Model chapter later in this document.

For information about IPv6, see the *IP Addresses and Services Configuration Guide for Cisco ASR 9000 Series Routers*.

CHAPTER 3

Ethernet Features

This chapter describes how to configure Layer 2 (L2) Ethernet features on the Cisco ASR 9000 Series Aggregation Services Routers supporting Cisco IOS XR software.

For more information on configuring Ethernet interfaces, refer to The Carrier Ethernet Model module of this configuration guide.

Feature History for Configuring Ethernet Interfaces on the Cisco ASR 9000 Series Routers

Release	Modification
Release 3.9.1	Support for Policy Based Forwarding and Layer 2 Protocol Tunneling features was added..

Prerequisites for Implementing Ethernet Features

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

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

Information About Implementing Ethernet Features

To configure 10-Gigabit Ethernet interfaces, you must understand these concepts:

Policy Based Forwarding

The Cisco ASR 9000 Series Routers allow a single MAC address to be mapped to a VLAN that is different from the port's configured VLAN. To separate the traffic entering two different EFPs, you must define an EFP using the source VLAN tag and the source MAC address.

Note This feature is supported only in the ASR 9000 Ethernet Line Card.

Layer 2 Protocol Tunneling

Layer 2 Protocol Tunneling (L2PT) is a Cisco proprietary protocol for tunneling Ethernet protocol frames across Layer 2 (L2) switching domains.

When an L2 protocol frame enters the interface of an L2 switching device, the switch or router performs one of these actions on the frame:

- forward—the frame is switched or routed with no exceptional handling.
- drop—the frame is discarded on the router.
- terminate—the router recognizes that the frame is an L2 protocol frame, and therefore sends it to the router's control plane for protocol processing.
- tunnel—the router encapsulates the frame to hide its identity as a protocol frame. This prevents the frame from being terminated on other routers. The opposite end of the tunnel performs a decapsulation, returning the frame to its original state.

L2PT Features

The Cisco ASR 9000 Series Routers offer these functions:

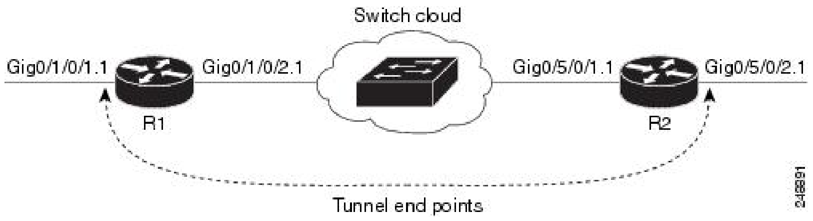
- Tunnels these protocols:
  - Cisco Discovery Protocol (CDP)
  - Spanning Tree Protocol (STP and its derivatives)
  - Virtual Trunking Protocol (VTP)
- Supports these modes of tunneling
  - Forward
  - Reverse
- L2PT encapsulates and decapsulates protocol frames that have VLAN headers.
- Supports capability of handling enormous frame rates. The Cisco ASR 9000 Series Routers perform L2PT encapsulation and decapsulation at the interface line rates.

Note There are no dedicated L2PT counters. There are no L2PT-specific adjustments for QoS or other miscellaneous parameters.

L2PT in the Forward Mode

Figure below shows L2PT configured in the forward mode.

Figure 4: L2PT in forward mode



A Service Provider network (S-network) is depicted in Figure 1. The customer network (C-network) connects to router R1 at the GigabitEthernet subinterface 0/1/0/1.1, and to router R2 at the GigabitEthernet subinterface 0/5/0/2.1. The C-network is not shown in the diagram; however, the C-network sends L2 traffic through the S-network, and the S-network switches the traffic from end to end. The customer traffic also carries L2 protocol frames. The purpose of L2PT is to allow these protocol frames to pass through the S-network. In forward mode, L2PT is applied to the customer facing interfaces of the S-network, R1 GigabitEthernet 0/1/0/1.1 and R2 GigabitEthernet 0/5/0/2.1.

Figure above depicts the configuration for L2PT in forward mode:

In the following examples, the **l2protocol cpsv tunnel** command is used for backward compatibility with legacy devices.

Note The **l2protocol cpsv drop** command is not supported on Cisco ASR 9000 series routers.

R1:

```
!
interface GigabitEthernet0/1/0/1
 negotiation auto
!
interface GigabitEthernet0/1/0/1.1 l2transport
 encapsulation default
 l2protocol cpsv tunnel
!
interface GigabitEthernet0/1/0/2
 negotiation auto
!
interface GigabitEthernet0/1/0/2.1 l2transport
 encapsulation default
!
l2vpn
 xconnect group examples
  p2p r1-connect
   interface GigabitEthernet0/1/0/1.1
   interface GigabitEthernet0/1/0/2.1
  !
 !
!
```

R2:

```
!
interface GigabitEthernet0/5/0/1
 negotiation auto
!
interface GigabitEthernet0/5/0/1.1 l2transport
 encapsulation default
```

```
!
interface GigabitEthernet0/5/0/2
 negotiation auto
!
interface GigabitEthernet0/5/0/2.1 l2transport
 encapsulation default
 l2protocol cpsv tunnel
!
l2vpn
 xconnect group examples
  p2p r2-connect
   interface GigabitEthernet0/5/0/1.1
   interface GigabitEthernet0/5/0/2.1
  !
 !
!
```

Protocol traffic enters router R1 at the GigabitEthernet subinterface 0/1/0/1.1. Router R1 detects the frames as protocol frames, and performs L2PT encapsulation at the customer facing interface. Inside R1, the local connection *r1-connect* connects R1's customer-facing and service provider-facing interfaces. The traffic then flows out of router R1 on GigabitEthernet subinterface 0/1/0/2.1 through several other service provider network routers or switches (switch cloud) into router R2 at GigabitEthernet subinterface 0/5/0/1.1. Router R2 connects the customer-facing and service provider-facing interfaces through a local connection *r2-connect.* Therefore, traffic is sent to the customer-facing interface GigabitEthernet 0/5/0/2.1. At this interface, an L2PT decapsulation occurs and the protocol traffic flows out of router R2 into the customer network.

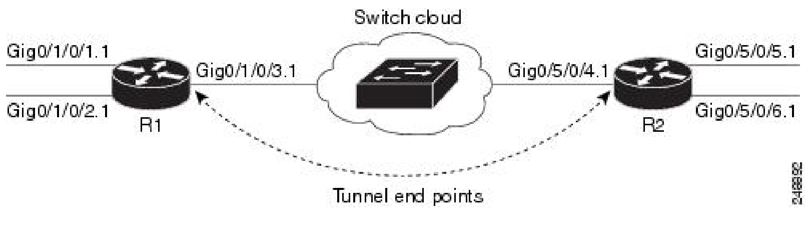
Without L2PT being configured the customer protocol frames that are sent into R1 are terminated. The customer traffic can consist of a variety of traffic; the protocol frames comprise a small percentage of the overall traffic stream.

L2PT in the Reverse Mode with Protocol Frame Tagging

The Cisco ASR 9000 Series Routers can perform L2PT encapsulation and decapsulation on supported L2 protocol frames that have VLAN headers. The L2 protocol frames do not have VLAN headers. However, in a service provider (SP) network that transports customer protocol traffic from one customer campus to another, this capability can be put to use within the SP network.

Figure below shows L2PT configured in the reverse mode. Assume that the customer traffic that enters R1 is trunked, that is all traffic is tagged. The only untagged traffic is the protocol traffic, that comes from the customer network.

Figure 5: L2PT in reverse mode



When L2PT is configured in the reverse mode, the L2PT encapsulation occurs when the frame exits the interface. Likewise, in reverse mode decapsulation is performed when the frame enters the interface. Therefore, the L2PT tunnel is formed between the service provider-facing interfaces, instead of the customer-facing interfaces.

In this example, once the protocol traffic enters router R1, a VLAN tag is added to it. Before the traffic is sent through the service provider network, a second VLAN tag is added (100). The Cisco ASR 9000 Series Routers perform the L2PT encapsulation on a double-tagged protocol frame.

The above figure above shows four customer-facing interfaces (R1: GigabitEthernet subinterface 0/1/0.1.1, GigabitEthernet subinterface 0/1/0/2.1 and R2: GigabitEthernet subinterface 0/5/0/5.1, GigabitEthernet subinterface 0/5/0/6.1) and two service provider-facing interfaces (R1: GigabitEthernet subinterface 0/1/0/3.1 and R2: GigabitEthernet subinterface 0/5/0/4.1).

Figure above depicts the configuration for L2PT in reverse mode:

At R1:

```
!
interface GigabitEthernet0/1/0/1
 negotiation auto
!
interface GigabitEthernet0/1/0/1.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 100 symmetric
 ethernet egress-filter strict
!
interface GigabitEthernet0/1/0/2
 negotiation auto
!
interface GigabitEthernet0/1/0/2.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 200 symmetric
 ethernet egress-filter strict
!
interface GigabitEthernet0/1/0/3
 negotiation auto
!
interface GigabitEthernet0/1/0/3.1 l2transport
 encapsulation dot1q 500
 rewrite ingress tag pop 1 symmetric
 l2protocol cpsv reverse-tunnel
 ethernet egress-filter strict
!
l2vpn
 bridge group examples
  bridge-domain r1-bridge
   interface GigabitEthernet0/1/0/1.1
   !
   interface GigabitEthernet0/1/0/2.1
!
   interface GigabitEthernet0/1/0/3.1
   !
  !
 !
!
```

At R2:

```
!
interface GigabitEthernet0/5/0/4
 negotiation auto
!
interface GigabitEthernet0/5/0/4.1 l2transport
 encapsulation dot1q 500
 rewrite ingress tag pop 1 symmetric
 l2protocol cpsv reverse-tunnel
 ethernet egress-filter strict
```

```
!
interface GigabitEthernet0/5/0/5
 negotiation auto
!
interface GigabitEthernet0/5/0/5.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 100 symmetric
 ethernet egress-filter strict
!
interface GigabitEthernet0/5/0/6
 negotiation auto
!
interface GigabitEthernet0/5/0/6.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 200 symmetric
 ethernet egress-filter strict
!
l2vpn
 bridge group examples
  bridge-domain r2-bridge
   interface GigabitEthernet0/5/0/4.1
   !
   interface GigabitEthernet0/5/0/5.1
   !
   interface GigabitEthernet0/5/0/6.1
   !
  !
 !
!
```

These assumptions are made:

- Customer traffic entering router R1 is trunked, that is all traffic is tagged. The only untagged traffic is the protocol traffic, which arrives from the customer network.
- The Customer-facing interfaces GigabitEthernet 0/1/0/1 at router R1 and Gigabit Ethernet 0/5/0/5 at router R2 belong to the same customer. Customer-facing interfaces GigabitEthernet 0/1/0/2 at router R1 and GigabitEthernet 0/5/0/6 at router R2 belong to a different customer.
- Traffic from different customers remain segregated.
- Only L2 protocol traffic is sent through the customer-facing interfaces.
- L2 protocol traffic entering the customer-facing interfaces is untagged.
- Traffic must be L2PT encapsulated to successfully pass through the switch cloud.

The purpose of this topology is that router R1 and R2 must receive customer protocol traffic from multiple customer interfaces, and multiplex the traffic across a single service provider interface and link. At the decapsulation end, the reverse is performed. Traffic entering router R1 on the GigabitEthernet subinterface 0/1/0/1.1 exits router R2 from the GigabitEthernet subinterface 0/5/0/5.1 only while traffic entering router R1 at GigabitEthernet subinterface 0/1/0/2.1 exits router R2 from GigabitEthernet subinterface 0/5/0/6.1 only.

A protocol frame entering router R1 on GigabitEthernet interface 0/1/0/1 travels through the network in this manner:

- The protocol frame is directed to GigabitEthernet subinterface 0/1/0/1.1, as the frame is untagged.
- The rewrite statement with GigabitEthernet subinterface 0/1/0/1.1 causes a tag of ID 100 to be added to the frame.

- The frame enters router R1's bridge domain r1-bridge.
- The bridge (r1-bridge) floods the frame to all attachment circuits (AC) on the bridge domain, except the originating AC (split horizon AC).
- Ethernet egress filtering on GigabitEthernet subinterface 0/1/0/2.1 detects a tag ID mismatch, and drops the frame. In this way, the bridge domain's flooded traffic is prevented from exiting other customer interfaces.
- A flooded copy of the frame is sent to GigabitEthernet subinterface 0/1/0/3.1.
- GigabitEthernet subinterface 0/1/0/3.1 adds a second tag.
- The frame receives an L2PT encapsulation by GigabitEthernet subinterface 0/1/0/3.1 before it leaves router R1 through the GigabitEthernet interface 0/1/0/3.

Note The frame is now double-tagged (100 inner, 500 outer) and has the L2PT MAC DA.

- The frame passes to router R2 GigabitEthernet interface 0/5/0/4 because of the L2PT encapsulation.
- The frame after having entered router R2 on GigabitEthernet interface 0/5/0/4 is directed to GigabitEthernet subinterface 0/5/0/4.1.
- On entering GigabitEthernet subinterface 0/5/0/4.1, an L2PT decapsulation operation is performed on the frame.
- The outer tag ID 500 is removed by GigabitEthernet subinterface 0/5/0/4.1
- Router R2's bridge (r2-bridge) floods the frames to all ACs.
- Ethernet egress filtering drops the frames on all ACs except the AC through which the frame exits.
- As the frame exits router R2 from GigabitEthernet subinterface 0/5/0/5.1, the tag of ID 100 is removed.
- The frame that exits router R2 from GigabitEthernet interface 0/5/0/5 is identical to the original frame that entered router R1 through GigabitEthernet interface 0/1/0/1.

L2PT Configuration Notes

Keep these points in mind while configuring L2PT:

- The **l2protocol** command can be configured on either a main or L2 subinterface.
- The **l2protocol** command can be configured on physical or bundle interfaces.
- When the **l2protocol** and **ethernet filtering** commands are configured on the same interface, L2PT encapsulation occurs before ethernet filtering. This means that L2PT prevents the CDP, STP, and VTP protocol frames from being dropped by ethernet filtering.
- When L2PT is configured with other interface features, L2PT encapsulation occurs before the processing for other interface features.

- L2PT encapsulation and decapsulation is supported for untagged protocol frames, single-tagged, and double-tagged frames. Tag Ethertypes of 0x8100, 0x88A8, and 0x9100 are supported, however, 0x9200 is not.

How to Implement Ethernet Features

Note For information on configuring Ethernet interfaces, refer to the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide.*

Restrictions for Implementing Ethernet Features

To use the sub-interface configurations **encapsulation dot1ad** (or **encapsulation dot1q**) and **encapsulation dot1ad second-dot1q any** (or **encapsulation dot1q second-dot1q any**) together on the same physical interface, use the **exact** keyword as shown below. Else, it results in traffic loss.

```
Router(config)# interface hundredGigE 0/0/0/0.0
Router(config-subif)# encapsulation dot1ad 200 exact
```

```
Router(config)# interface hundredGigE 0/0/0/0.1
Router(config-subif)# encapsulation dot1ad 200 dot1q any
Router(config-subif)# commit
```

Configuring Policy Based Forwarding

Enabling Policy Based Forwarding

Perform this task to enable policy based forwarding.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id.subinterface* **l2transport**
3. Do one of the following:
   - **encapsulation dot1q** *vlan-id* **ingress source-mac** *mac-address* or
   - **encapsulation dot1ad** *vlan-id* **ingress source-mac** *mac-address* or
   - **encapsulation untagged ingress source-mac** *mac-address* or
   - **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* **ingress source-mac** *mac-address*
4. Do one of the following:
   - **rewrite ingress tag translate 1-to-1 dot1q** *vlan-id* **symmetric** or
   - **rewrite tag push dot1q** *vlan-id* **symmetric**
5. **ethernet egress-filter strict**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **interface** *type interface-path-id.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router# interface Bundle-Ether1.1 l2transport
```

Enters subinterface configuration mode and enables Layer 2 transport mode on a port and enters Layer 2 transport configuration mode.

Step 3 Do one of the following:

- **encapsulation dot1q** *vlan-id* **ingress source-mac** *mac-address* or
- **encapsulation dot1ad** *vlan-id* **ingress source-mac** *mac-address* or
- **encapsulation untagged ingress source-mac** *mac-address* or
- **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* **ingress source-mac** *mac-address*

Example:

```
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1q 10 ingress source-mac 0.1.2
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1ad 10 ingress source-mac 0.1.4
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation untagged ingress source-mac 0.1.3
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1ad 10 dot1q 10 ingress source-mac 0.1.2
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1q 10 second-dot1q 20 ingress source-mac 0.1.2
```

Assigns the matching VLAN ID and Ethertype to the interface.

Note The **encapsulation dot1q** *vlan-id* **ingress source-mac** *mac-address* command is supported only on bundle sub-interfaces.

Step 4 Do one of the following:

- **rewrite ingress tag translate 1-to-1 dot1q** *vlan-id* **symmetric** or
- **rewrite tag push dot1q** *vlan-id* **symmetric**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# rewrite ingress tag translate 1-to-1 dot1q 100 symmetric
or
rewrite ingress tag push dot1q 101 symmetric
```

Specifies the encapsulation adjustment that is to be performed on the frame ingress to the service instance.

Step 5 **ethernet egress-filter strict**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# ethernet egress-filter strict
```

Enables strict egress filtering on all subinterfaces.

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

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

end - Prompts user to take one of these actions:

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

Configuring Source Bypass Filter

Perform this task to add a source bypass filter.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id.subinterface* **l2transport**
3. Do one of the following:
   - **encapsulation dot1q** *vlan-id* or
   - **encapsulation dot1ad** *vlan-id* or
   - **encapsulation untagged** or
   - **encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id* or
   - **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* or
4. **rewrite ingress tag translate translate 1-to-1 dot1q** *vlan-id* **symmetric**
5. **ethernet egress-filter disable**
6. **ethernet source bypass egress-filter**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **interface** *type* *interface-path-id.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet 0/2/0/4.1 l2transport
```

Enters subinterface configuration mode and enables Layer 2 transport mode on a port and enters Layer 2 transport configuration mode.

Step 3 Do one of the following:

- **encapsulation dot1q** *vlan-id* or
- **encapsulation dot1ad** *vlan-id* or
- **encapsulation untagged** or
- **encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id* or
- **encapsulation dot1q** *vlan-id* **second-dot1q** *vlan-id* or

Example:

```
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1q 10
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1ad 10
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation untagged
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1ad 10 dot1q 10
or
RP/0/RSP0/CPU0:router(config-subif)#
encapsulation dot1q 10 second-dot1q 20
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 4 **rewrite ingress tag translate translate 1-to-1 dot1q** *vlan-id* **symmetric**

Example:

```
RP/0/RSP0/CPU0:router
(config-subif)# rewrite ingress tag translate 1-to-1 dot1q 100 symmetric
```

Specifies the encapsulation adjustment that is to be performed on the frame ingress to the service instance.

Step 5 **ethernet egress-filter disable**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# ethernet egress-filter strict
```

Disables egress filtering on all subinterfaces.

Step 6 **ethernet source bypass egress-filter**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# ethernet source bypass egress-filter
```

Enables source bypass egress filtering on the subinterfaces.

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

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

end - Prompts user to take one of these actions:

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

Configuration Examples

Configuring Policy Based Forwarding: Example

This example shows how to configure policy based forwarding:

```
config
 interface Bundle-Ether1.1 l2transport
 encapsulation dot1q 10 ingress source-mac 0000.1111.2222
 rewrite ingress tag translate 1-to-1 dot1q 100 symmetric
 ethernet egress-filter strict
!
interface Bundle-Ether1.2 l2transport
 encapsulation untagged ingress source-mac 0000.1111.3333
 rewrite ingress tag push dot1q 101 symmetric
 ethernet egress-filter strict
!

interface Bundle-Ether3.1 l2transport
 encapsulation dot1q 1
 rewrite ingress tag translate 1-to-1 dot1q 4094 symmetric
 ethernet egress-filter disabled
 ethernet source-bypass-egress-filter
!
```

Configuring Layer 2 Protocol Tunneling: Example

This section includes configuration examples for L2PT in the forward and reverse modes.

Configuring L2PT in forward mode

This example shows how to configure L2PT in the forward mode. In the following examples, the **l2protocol cpsv tunnel** command is used for backward compatibility with legacy devices.

Note The **l2protocol cpsv drop** command is not supported on Cisco ASR 9000 series routers.

At the customer facing router (encapsulation end):

```
!
interface GigabitEthernet0/1/0/1
 negotiation auto
```

```
!
interface GigabitEthernet0/1/0/1.1 l2transport
 encapsulation default
 l2protocol cpsv tunnel
!
interface GigabitEthernet0/1/0/2
 negotiation auto
!
interface GigabitEthernet0/1/0/2.1 l2transport
 encapsulation default
!
l2vpn
 xconnect group examples
  p2p r1-connect
   interface GigabitEthernet0/1/0/1.1
   interface GigabitEthernet0/1/0/2.1
  !
 !
!
```

At the customer facing router (decapsulation end):

```
!
interface GigabitEthernet0/5/0/1
 negotiation auto
!
interface GigabitEthernet0/5/0/1.1 l2transport
 encapsulation default
!
interface GigabitEthernet0/5/0/2
 negotiation auto
!
interface GigabitEthernet0/5/0/2.1 l2transport
 encapsulation default
 l2protocol cpsv tunnel
!
l2vpn
 xconnect group examples
  p2p r2-connect
   interface GigabitEthernet0/5/0/1.1
   interface GigabitEthernet0/5/0/2.1
  !
 !
!
```

Configuring L2PT in reverse mode

This example shows how to configure L2PT in the reverse mode:

At the customer facing router (encapsulation end):

```
!
interface GigabitEthernet0/1/0/1
 negotiation auto
!
interface GigabitEthernet0/1/0/1.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 100 symmetric
 ethernet egress-filter strict
!
interface GigabitEthernet0/1/0/2
 negotiation auto
!
interface GigabitEthernet0/1/0/2.1 l2transport
```

```
 encapsulation untagged
 rewrite ingress tag push dot1q 200 symmetric
 ethernet egress-filter strict
!
interface GigabitEthernet0/1/0/3
 negotiation auto
!
interface GigabitEthernet0/1/0/3.1 l2transport
 encapsulation dot1q 500
 rewrite ingress tag pop 1 symmetric
 l2protocol cpsv reverse-tunnel
 ethernet egress-filter strict
!
l2vpn
 bridge group examples
  bridge-domain r1-bridge
   interface GigabitEthernet0/1/0/1.1
   !
   interface GigabitEthernet0/1/0/2.1
   !
   interface GigabitEthernet0/1/0/3.1
   !
  !
 !
!
```

At the customer facing router (decapsulation end):

```
!
interface GigabitEthernet0/5/0/4
 negotiation auto
!
interface GigabitEthernet0/5/0/4.1 l2transport
 encapsulation dot1q 500
 rewrite ingress tag pop 1 symmetric
 l2protocol cpsv reverse-tunnel
 ethernet egress-filter strict
!
interface GigabitEthernet0/5/0/5
 negotiation auto
!
interface GigabitEthernet0/5/0/5.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 100 symmetric
 ethernet egress-filter strict
!
interface GigabitEthernet0/5/0/6
 negotiation auto
!
interface GigabitEthernet0/5/0/6.1 l2transport
 encapsulation untagged
 rewrite ingress tag push dot1q 200 symmetric
 ethernet egress-filter strict
!
l2vpn
 bridge group examples
  bridge-domain r2-bridge
   interface GigabitEthernet0/5/0/4.1
   !
   interface GigabitEthernet0/5/0/5.1
   !
   interface GigabitEthernet0/5/0/6.1
   !
  !
  !
 !
```

CHAPTER 4

Configuring Link Bundles

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The different links within a single bundle can have varying speeds, where the fastest link can be a maximum of four times greater than the slowest link. Each bundle has a single MAC, a single IP address, and a single configuration set (such as ACLs or QoS).

The router supports bundling for these types of interfaces:

- Ethernet interfaces
- VLAN subinterfaces

Note Bundles do not have a one-to-one modular services card association.

Feature History for Configuring Link Bundles

Release	Modification
Release 3.7.2	This feature was introduced on the Cisco ASR 9000 Series Routers.

Prerequisites for Configuring Link Bundles

Before configuring Link Bundling, be sure that these tasks and conditions are met:

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

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

- You know the interface IP address.
- You know which links should be included in the bundle you are configuring.
- If you are configuring an Ethernet link bundle, you have at least one of these Ethernet line cards installed in the router:
    - 2-port 10-Gigabit Ethernet line card
    - 4-port 10-Gigabit Ethernet line card
    - 8-port 10-Gigabit Ethernet line card
    - 16-port 10-Gigabit Ethernet line card
    - 20-port Gigabit Ethernet line card
    - 40-port Gigabit Ethernet line card

Note For more information about physical interfaces, PLIMs, and modular services cards, refer to the *Cisco ASR 9000 Series Aggregation Services Router Hardware Installation Guide* .

Information About Configuring Link Bundles

To implement the Link Bundling feature, you must understand these concepts:

Link Bundling Overview

A link bundle is simply a group of ports that are bundled together and act as a single link. The advantages of link bundles are these:

- Multiple links can span several line cards to form a single interface. Thus, the failure of a single link does not cause a loss of connectivity.
- Bundled interfaces increase bandwidth availability, because traffic is forwarded over all available members of the bundle. Therefore, traffic can flow on the available links if one of the links within a bundle fails. Bandwidth can be added without interrupting packet flow.

Although the individual links within a single bundle can have varying speeds, all links within a bundle must be of the same type.

Cisco IOS XR software supports these methods of forming bundles of Ethernet interfaces:

- IEEE 802.3ad—Standard technology that employs a Link Aggregation Control Protocol (LACP) to ensure that all the member links in a bundle are compatible. Links that are incompatible or have failed are automatically removed from a bundle.
- EtherChannel—Cisco proprietary technology that allows the user to configure links to join a bundle, but has no mechanisms to check whether the links in a bundle are compatible.

Characteristics of Link Bundles

This list describes the properties and limitations of link bundles:

- Any type of Ethernet interfaces can be bundled, with or without the use of LACP (Link Aggregation Control Protocol).
- Bundle membership can span across several line cards that are installed in a single router.
- A single bundle supports maximum of 64 physical links.
- Different link speeds are allowed within a single bundle, with a maximum of four times the speed difference between the members of the bundle.
- Physical layer and link layer configuration are performed on individual member links of a bundle.
- Configuration of network layer protocols and higher layer applications is performed on the bundle itself.
- A bundle can be administratively enabled or disabled.
- Each individual link within a bundle can be administratively enabled or disabled.
- Ethernet link bundles are created in the same way as Ethernet channels, where the user enters the same configuration on both end systems.
- The MAC address that is set on the bundle becomes the MAC address of the links within that bundle.
- When LACP configured, each link within a bundle can be configured to allow different keepalive periods on different members.
- Load balancing (the distribution of data between member links) is done by flow instead of by packet. Data is distributed to a link in proportion to the bandwidth of the link in relation to its bundle.
- QoS is supported and is applied proportionally on each bundle member.
- Link layer protocols, such as CDP and HDLC keepalives, work independently on each link within a bundle.
- Upper layer protocols, such as routing updates and hellos, are sent over any member link of an interface bundle.
- Bundled interfaces are point to point.
- A link must be in the up state before it can be in distributing state in a bundle.
- All links within a single bundle must be configured either to run 802.3ad (LACP) or Etherchannel (non-LACP). Mixed links within a single bundle are not supported.
- A bundle interface can contain physical links and VLAN subinterfaces only.
- Access Control List (ACL) configuration on link bundles is identical to ACL configuration on regular interfaces.
- Multicast traffic is load balanced over the members of a bundle. For a given flow, internal processes select the member link and all traffic for that flow is sent over that member.

IEEE 802.3ad Standard

The IEEE 802.3ad standard typically defines a method of forming Ethernet link bundles.

For each link configured as bundle member, this information is exchanged between the systems that host each end of the link bundle:

- A globally unique local system identifier
- An identifier (operational key) for the bundle of which the link is a member
- An identifier (port ID) for the link
- The current aggregation status of the link

This information is used to form the link aggregation group identifier (LAG ID). Links that share a common LAG ID can be aggregated. Individual links have unique LAG IDs.

The system identifier distinguishes one router from another, and its uniqueness is guaranteed through the use of a MAC address from the system. The bundle and link identifiers have significance only to the router assigning them, which must guarantee that no two links have the same identifier, and that no two bundles have the same identifier.

The information from the peer system is combined with the information from the local system to determine the compatibility of the links configured to be members of a bundle.

Bundle MAC addresses in the routers come from a set of reserved MAC addresses in the backplane. This MAC address stays with the bundle as long as the bundle interface exists. The bundle uses this MAC address until the user configures a different MAC address. The bundle MAC address is used by all member links when passing bundle traffic. Any unicast or multicast addresses set on the bundle are also set on all the member links.

Note We recommend that you avoid modifying the MAC address, because changes in the MAC address can affect packet forwarding.

Non Revertive Behavior for LACP Bundle Interface

In LACP, by default, a higher priority port would become the active port after it becomes operational again. To avoid this reversion, you can run the **lacp non-revertive** command. This configures the lower priority port to continue as the active port even after the higher priority port is capable of being operational. This avoids the traffic disruption that may happen in putting the currently active but lower priority port into standby and diverting traffic through the higher priority port that is now capable of being operational.

QoS and Link Bundling

On the ingress direction, QoS is applied to the local instance of a bundle. Each bundle is associated with a set of queues. QoS is applied to the various network layer protocols that are configured on the bundle.

On the egress direction, QoS is applied on the bundle with a reference to the member links. QoS is applied based on the sum of the member bandwidths.

When QoS is applied on the bundle for either the ingress or egress direction, QoS is applied at each member interface.

The Link Bundling feature supports all the QoS features described in the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Configuration Guide*.

The Link Bundling feature supports these QoS features:

- hi priority /lo priority—Maximum bandwidth is calculated as a percentage of the bundle interface bandwidth. This percentage is then applied to every member link on the egress, or to the local bundle instance on ingress.
- guaranteed bandwidth—Provided in percentage and applied to every member link.
- traffic shaping—Provided in percentage and applied to every member link.
- WRED—Minimum and maximum parameters are converted to the right proportion per member link or bundle instance, and then are applied to the bundle.
- marking—Process of changing the packet QoS level according to a policy.
- tail drop— Packets are dropped when the queue is full.

VLANs on an Ethernet Link Bundle

802.1Q VLAN subinterfaces can be configured on 802.3ad Ethernet link bundles. Keep this information in mind when adding VLANs on an Ethernet link bundle:

- The maximum number of VLANs allowed per bundle is 4000.
- The maximum number of bundled VLANs allowed per router is 128000.

Note The memory requirement for bundle VLANs is slightly higher than standard physical interfaces.

To create a VLAN subinterface on a bundle, include the VLAN subinterface instance with the **interface Bundle-Ether** command:

interface Bundle-Ether `instance.subinterface`

After you create a VLAN on an Ethernet link bundle, all physical VLAN subinterface configuration is supported on that link bundle.

Link Bundle Configuration Overview

These steps provide a general overview of the link bundle configuration process. Keep in mind that a link must be cleared of all previous network layer configuration before it can be added to a bundle:

1. In global configuration mode, create a link bundle. To create an Ethernet link bundle, enter the **interface Bundle-Ether** command.
2. Assign an IP address and subnet mask to the virtual interface using the **ipv4 address** command.
3. Add interfaces to the bundle you created in Step 1 with the **bundle id** command in the interface configuration submode. You can add up to 32 links to a single bundle.

Note A link is configured to be a member of a bundle from the interface configuration submode for that link.

Nonstop Forwarding During Card Failover

The Cisco IOS XR software supports nonstop forwarding during failover between active and standby paired RSP cards. Nonstop forwarding ensures that there is no change in the state of the link bundles when a failover occurs.

For example, if an active RSP fails, the standby RSP becomes operational. The configuration, node state, and checkpoint data of the failed RSP are replicated to the standby RSP. The bundled interfaces will all be present when the standby RSP becomes the active RSP.

Note Failover is always onto the standby RSP.

Note You do not need to configure anything to guarantee that the standby interface configurations are maintained.

Link Failover

When one member link in a bundle fails, traffic is redirected to the remaining operational member links and traffic flow remains uninterrupted.

Bundle Interfaces: Redundancy, Load Sharing, Aggregation

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The different links within a single bundle can have varying speeds, where the fastest link can be a maximum of four times greater than the slowest link. Each bundle has a single MAC, a single IP address, and a single configuration set (such as ACLs or QoS).

The router supports bundling for these types of interfaces:

- Ethernet interfaces
- VLAN subinterfaces

How to Configure Link Bundling

Configuring Ethernet Link Bundles

This section describes how to configure a Ethernet link bundle.

Note MAC accounting is not supported on Ethernet link bundles.

Note In order for an Ethernet bundle to be active, you must perform the same configuration on both connection endpoints of the bundle.

The creation of an Ethernet link bundle involves creating a bundle and adding member interfaces to that bundle, as shown in the steps that follow.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address-address mask*
4. **bundle minimum-active bandwidth** *kbps* (optional)
5. **bundle minimum-active links** *links* (optional)
6. **bundle maximum-active links** *links* (optional)
7. **bundle maximum-active links** *links* **hot-standby**
8. **exit**
9. **interface {GigabitEthernet | TenGigE}** *instance*
10. **bundle id** *bundle-id* [**mode** { **active** | **on** | **passive**}]
11. **no shutdown**(optional)
12. **exit**
13. Repeat Step 8 through Step 11 to add more links to the bundle you created in Step 2.
14. Use the **commit** or **end** command.
15. **exit**
16. **exit**
17. Perform Step 1 through Step 15 on the remote end of the connection.
18. **show bundle Bundle-Ether** *bundle-id* [**reasons**] (optional)
19. **show lacp Bundle-Ether** *bundle-id* (optional)

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface specific configuration commands are entered. Use the **exit** command to exit from the interface configuration submode back to the normal Global Configuration mode.

Step 3 **ipv4 address** *ipv4-address-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps* (optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle minimum-active bandwidth 580000
```

Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links* (optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle minimum-active links 2
```

Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* (optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle maximum-active links 1
```

Designates one active link and one link in standby mode that can take over immediately for a bundle if the active link fails (1:1 protection).

The default number of active links allowed in a single bundle is 8.

Note If the **bundle maximum-active** command is issued, then only the highest-priority link within the bundle is active. The priority is based on the value from the **bundle port-priority** command, where a lower value is a higher priority. Therefore, we recommend that you configure a higher priority on the link that you want to be the active link.

Step 7 **bundle maximum-active links** *links* **hot-standby**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle maximum-active links 1 hot-standby
```

The **hot-standby** keyword helps to avoid bundle flaps on a switchover or switchback event during which the bundle temporarily falls below the minimum links or bandwidth threshold.

It sets default values for the wait-while timer and suppress-flaps timer to achieve this.

Step 8 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-#if)exit
```

Exits interface configuration submode for the Ethernet link bundle.

Step 9 **interface {GigabitEthernet | TenGigE}** *instance*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE 1/0/0/0
```

Enters the interface configuration mode for the specified interface.

Enter the **GigabitEthernet** or **TenGigE** keyword to specify the interface type. Replace the instance argument with the node-id in the *rack/slot/module* format.

Mixed bandwidth bundle member configuration is only supported when 1:1 redundancy is configured (this means that a 1 GigabitEthernet member can only be configured as the backup of the 10 GigabitEthernet interface.)

Note Mixed link bundle mode is supported only when active-standby operation is configured (usually with the lower speed link in standby mode).

Step 10 **bundle id** *bundle-id* [**mode** { **active** | **on** | **passive**}]

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle-id 3
```

Adds the link to the specified bundle.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the link to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note If you do not specify the **mode** keyword, the default mode is **on** (LACP is not run over the port).

Step 11 **no shutdown**(optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# no shutdown
```

If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# exit
```

Exits interface configuration submode for the Ethernet link bundle.

Step 13 Repeat Step 8 through Step 11 to add more links to the bundle you created in Step 2.

Step 14 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.

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

Step 15 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# exit
```

Exits interface configuration mode.

Step 16 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config)# exit
```

Exits the Global Configuration mode.

Step 17 Perform Step 1 through Step 15 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 18 **show bundle Bundle-Ether** *bundle-id* [**reasons**] (optional)

Example:

```
RP/0/RSP0/CPU0:router# show bundle Bundle-Ether 3 reasons
```

Shows information about the specified Ethernet link bundle

Step 19 **show lacp Bundle-Ether** *bundle-id* (optional)

Example:

```
RP/0/RSP0/CPU0:router # show lacp Bundle-Ether 3
```

Shows detailed information about LACP ports and their peers.

Configuring VLAN Bundles

This section describes how to configure a VLAN bundle. The creation of a VLAN bundle involves three main tasks:

1. Create an Ethernet bundle.
2. Create VLAN subinterfaces and assign them to the Ethernet bundle.
3. Assign Ethernet links to the Ethernet bundle.

These tasks are describe in detail in the procedure that follows.

Note In order for a VLAN bundle to be active, you must perform the same configuration on both ends of the bundle connection.

The creation of a VLAN link bundle is described in the steps that follow.

SUMMARY STEPS

1. **configure**
2. **interface Bundle-Ether** *bundle-id*
3. **ipv4 address** *ipv4-address mask*
4. **bundle minimum-active bandwidth** *kbps* (optional)
5. **bundle minimum-active links** *links* (optional)
6. **bundle maximum-active links** *links* (optional)
7. **exit**
8. **interface Bundle-Ether** *bundle-id.vlan-id*
9. **encapsulation dot1q** *vlan-id*
10. **ipv4 address** *ip-address mask*
11. **no shutdown**
12. **exit**
13. Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.
14. Use the **commit** or **end** command.
15. **exit**
16. **exit**
17. **show ethernet trunk bundle-Ether** *instance*
18. **configure**
19. **interface {GigabitEthernet | TenGigE}** *instance*
20. **bundle id** `bundle-id` **[mode {active | on | passive}]**
21. **no shutdown**
22. Repeat Step 19 through Step 21 to add more Ethernet interfaces to the bundle you created in Step 2 .
23. Use the **commit** or **end** command.
24. Perform Step 1 through Step 23 on the remote end of the connection.
25. **show bundle Bundle-Ether** *bundle-id* [**reasons**]
26. **show ethernet trunk bundle-Ether** *instance*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RSP0/CPU0:router#(config)# interface Bundle-Ether 3
```

Creates and names a new Ethernet link bundle.

This **interface Bundle-Ether** command enters you into the interface configuration submode, where you can enter interface specific configuration commands are entered. Use the **exit** command to exit from the interface configuration submode back to the normal Global Configuration mode

Step 3 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.1.2.3 255.0.0.0
```

Assigns an IP address and subnet mask to the virtual interface using the **ipv4 address** configuration subcommand.

Step 4 **bundle minimum-active bandwidth** *kbps* (optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if) # bundle minimum-active bandwidth 580000
```

(Optional) Sets the minimum amount of bandwidth required before a user can bring up a bundle.

Step 5 **bundle minimum-active links** *links* (optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle minimum-active links 2
```

(Optional) Sets the number of active links required before you can bring up a specific bundle.

Step 6 **bundle maximum-active links** *links* (optional)

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bundle maximum-active links 1
```

(Optional) Designates one active link and one link in standby mode that can take over immediately for a bundle if the active link fails (1:1 protection).

Note The default number of active links allowed in a single bundle is 8.

Note If the **bundle maximum-active** command is issued, then only the highest-priority link within the bundle is active. The priority is based on the value from the **bundle port-priority** command, where a lower value is a higher priority. Therefore, we recommend that you configure a higher priority on the link that you want to be the active link.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# exit
```

Exits interface configuration submode.

Step 8 **interface Bundle-Ether** *bundle-id.vlan-id*

Example:

```
RP/0/RSP0/CPU0:router#(config)#interface Bundle-Ether 3.1
```

Creates a new VLAN, and assigns the VLAN to the Ethernet bundle you created in Step 2.

Replace the *bundle-id* argument with the *bundle-id* you created in Step 2.

Replace the *vlan-id* with a subinterface identifier. Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Note When you include the *vlan-id* argument with the **interface Bundle-Ether** *bundle-id* command, you enter subinterface configuration mode.

Step 9 **encapsulation dot1q** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router#(config-subif)# encapsulation dot1q 10
```

Assigns a VLAN to the subinterface.

Replace the *vlan-id* argument with a subinterface identifier. Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Step 10 **ipv4 address** *ip-address mask*

Example:

```
RP/0/RSP0/CPU0:router#(config-subif)# ipv4 address 10.1.2.3/24
```

Assigns an IP address and subnet mask to the subinterface.

Step 11 **no shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-subif) # no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 12 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)#exit
```

Exits subinterface configuration mode for the VLAN subinterface.

Step 13 Repeat Step 7 through Step 12 to add more VLANs to the bundle you created in Step 2.

(Optional) Adds more subinterfaces to the bundle.

Step 14 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Step 15 **exit**

Example:

```
RP/0/RSP0/CPU0:router (config-subif)# exit
```

Exits interface configuration mode.

Step 16 **exit**

Example:

```
RP/0/RSP0/CPU0:router (config)# exit
```

Exits Global Configuration mode.

Step 17 **show ethernet trunk bundle-Ether** *instance*

Example:

```
RP/0/RSP0/CPU0:routerrouter# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Step 18 **configure**

Example:

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

Enters Global Configuration mode.

Step 19 **interface {GigabitEthernet | TenGigE}** *instance*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE 1/0/0/0
```

Enters the interface configuration mode for the specified interface.

Replace the *instance* argument with the node-id in the *rack/slot/module* format.

Note A VLAN bundle is not active until you add an Ethernet interface on both ends of the link bundle.

Step 20 **bundle id `bundle-id` [mode {active | on | passive}]**

Example:

```
RP/0/RSP0/CPU0:router(config-if)#  bundle-id 3
```

Adds an Ethernet interface to the bundle you configured in Step 2 through Step 13.

To enable active or passive LACP on the bundle, include the optional **mode active** or **mode passive** keywords in the command string.

To add the interface to the bundle without LACP support, include the optional **mode on** keywords with the command string.

Note If you do not specify the **mode** keyword, the default mode is **on** (LACP is not run over the port).

Step 21 **no shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# no shutdown
```

(Optional) If a link is in the down state, bring it up. The **no shutdown** command returns the link to an up or down state depending on the configuration and state of the link.

Step 22 Repeat Step 19 through Step 21 to add more Ethernet interfaces to the bundle you created in Step 2 .

Step 23 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Step 24 Perform Step 1 through Step 23 on the remote end of the connection.

Brings up the other end of the link bundle.

Step 25 **show bundle Bundle-Ether** *bundle-id* [**reasons**]

Example:

```
RP/0/RSP0/CPU0:router#show bundle Bundle-Ether 3 reasons
```

(Optional) Shows information about the specified Ethernet link bundle.

The **show bundle Bundle-Ether** command displays information about the specified bundle. If your bundle has been configured properly and is carrying traffic, the State field in the **show bundle Bundle-Ether** command output will show the number “4,” which means the specified VLAN bundle port is “distributing.”

Step 26 **show ethernet trunk bundle-Ether** *instance*

Example:

```
RP/0/RSP0/CPU0:router# show ethernet trunk bundle-ether 5
```

(Optional) Displays the interface configuration.

The Ethernet bundle instance range is from 1 through 65535.

Configuration Examples for Link Bundles

EtherChannel Bundle running LACP: Example

This example shows how to join two ports to form an EtherChannel bundle running LACP:

```
RP/0/RSP0/CPU0:Router# config
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 3
RP/0/RSP0/CPU0:Router(config-if)# ipv4 address 1.2.3.4/24
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active bandwidth 620000
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active links 1
RP/0/RSP0/CPU0:Router(config-if)# exit
RP/0/RSP0/CPU0:Router(config)# interface TenGigE 0/3/0/0
RP/0/RSP0/CPU0:Router(config-if)# bundle id 3 mode active
RP/0/RSP0/CPU0:Router(config-if)# no shutdown
RP/0/RSP0/CPU0:Router(config)# exit
RP/0/RSP0/CPU0:Router(config)# interface TenGigE 0/3/0/1
RP/0/RSP0/CPU0:Router(config-if)# bundle id 3 mode active
```

```
RP/0/RSP0/CPU0:Router(config-if)# no shutdown
RP/0/RSP0/CPU0:Router(config-if)# exit
```

Creating VLANs on a Ethernet Bundle: Example

This example shows how to create and bring up two VLANs on an Ethernet bundle:

```
RP/0/RSP0/CPU0:Router# config
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 1
RP/0/RSP0/CPU0:Router(config-if)# ipv4 address 1.2.3.4/24
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active bandwidth 620000
RP/0/RSP0/CPU0:Router(config-if)# bundle minimum-active links 1
RP/0/RSP0/CPU0:Router(config-if)# exit
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 1.1
RP/0/RSP0/CPU0:Router(config-subif)# encapsulation dot1q 10
RP/0/RSP0/CPU0:Router(config-subif)# ip addr 10.2.3.4/24
RP/0/RSP0/CPU0:Router(config-subif)# no shutdown
RP/0/RSP0/CPU0:Router(config-subif)# exit
RP/0/RSP0/CPU0:Router(config)# interface Bundle-Ether 1.2
RP/0/RSP0/CPU0:Router(config-subif)# encapsulation dot1q 20
RP/0/RSP0/CPU0:Router(config-subif)# ip addr 20.2.3.4/24
RP/0/RSP0/CPU0:Router(config-subif)# no shutdown
RP/0/RSP0/CPU0:Router(config-subif)# exit
RP/0/RSP0/CPU0:Router(config)#interface tengige 0/1/5/7
RP/0/RSP0/CPU0:Router(config-if)# bundle-id 1 mode act
RP/0/RSP0/CPU0:Router(config-if)# commit
RP/0/RSP0/CPU0:Router(config-if)# exit
RP/0/RSP0/CPU0:Router(config)# exit
RP/0/RSP0/CPU0:Router # show ethernet trunk bundle-ether 1
```

ASR 9000 Link Bundles connected to a Cisco 7600 EtherChannel: Example

This example is an end-to-end example of a bundle between ASR 9000 Series router (ASR-9010) and a Cisco 7600 Series Router (P19_C7609-S) in the Metro Ethernet network that supports both L2 and L3 services.

On the Cisco ASR 9000 Series Routers, the bundle is configured with LACP, 1:1 link protection, two L2 subinterfaces, and two layer 3 subinterfaces.

IOS XR side:

```
hostname PE44_IOS-XR_Router

interface Bundle-Ether16
 description Connect to P19_C7609-S Port-Ch 16
 mtu 9216
 no ipv4 address
 bundle maximum-active links 1
!
interface Bundle-Ether16.160 l2transport
 description Connect to P19_C7609-S Port-Ch 16 EFP 160
 encapsulation dot1q 160
!
interface Bundle-Ether16.161 l2transport
 description Connect to P19_C7609-S Port-Ch 16 EFP 161
 encapsulation dot1q 161
!
interface Bundle-Ether16.162
 description Connect to P19_C7609-S Port-Ch 16.162
 ipv4 address 10.194.8.44 255.255.255.0
 encapsulation dot1q 162
```

```
!
interface Bundle-Ether16.163
 description Connect to P19_C7609-S Port-Ch 16.163
 ipv4 address 10.194.12.44 255.255.255.0
 encapsulation dot1q 163
!


interface TenGigE 0/1/0/16
 description Connected to P19_C7609-S GE 8/0/16
 bundle id 16 mode active
 bundle port-priority 1
!
interface TenGigE 0/1/0/17
 description Connected to P19_C7609-S GE 8/0/17
 bundle id 16 mode active
 bundle port-priority 2
!
```

IOS XR side - connections to CE devices:

```
hostname PE44_IOS-XR_Router

interface TenGigE 0/1/0/3.160 l2transport
 description VLAN 160 over BE 16.160
 encapsulation dot1q 100 second-dot1q 160
 rewrite ingress tag pop 1 symmetric
!
interface TenGigE 0/1/0/3.161 l2transport
 description VLAN 161 over BE 16.161
 encapsulation dot1q 161
!
l2vpn
 !
 xconnect group 160
  p2p 160
   interface Bundle-Ether16.160
   interface TenGigE 0/1/0/3.160
   description VLAN_160_over_BE_16.160
  !
!
 xconnect group 161
  p2p 161
   interface Bundle-Ether16.161
   interface TenGigE 0/1/0/3.161
   description VLAN_161_over_BE_16.161
  !
 !
```

IOS XR side - CE devices:

```
hostname PE64_C3750-ME
!
vlan 161
!
interface TenGigE 1/0/1
 description Connected to PE65_ME-C3400 GE 0/1
 switchport access vlan 100
 switchport mode dot1q-tunnel
!
interface TenGigE 1/0/2
 description Connected to PE44_IOS-XR_Router GE 0/1/0/3
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 100,161
 switchport mode trunk
!
interface Vlan161
 description VLAN 161 over BE 16.161 on PE44
 ip address 161.0.0.64 255.255.255.0
!



hostname PE65_ME-C3400
!
vlan 160
!
interface TenGigE 0/1
 description Connected to PE64_C3750-ME GE 1/0/1
 port-type nni
 switchport trunk allowed vlan 160
 switchport mode trunk
!
interface Vlan160
 description VLAN 160 over BE 16.160 on PE44
 ip address 160.0.0.65 255.255.255.0
!
```

IOS side:

```
hostname P19_C7609-S

port-channel load-balance src-dst-port
!
interface Port-channel16
 description Connected to PE44_IOS-XR_Router BE 16
 mtu 9202
 no ip address
 logging event link-status
 logging event status
 speed nonegotiate
 mls qos trust dscp
 lacp fast-switchover
 lacp max-bundle 1
 service instance 160 ethernet
  description Connected to PE44_IOS-XR_Router BE 16.160
  encapsulation dot1q 160
 !
 service instance 161 ethernet
  description Connected to PE44_IOS-XR_Router BE 16.161
  encapsulation dot1q 161
 !
!
interface Port-channel16.162
 description Connected to PE44_IOS-XR_Router BE 16.162
 encapsulation dot1Q 162
 ip address 10.194.8.19 255.255.255.0
!
interface Port-channel16.163
 description Connected to PE44_IOS-XR_Router BE 16.163
 encapsulation dot1Q 163
 ip address 10.194.12.19 255.255.255.0
!

interface TenGigE 8/0/16
 no shut
 description Connected to PE44_IOS-XR_Router GE 0/1/0/16
 mtu 9202
 no ip address
 logging event link-status
 logging event status
 speed nonegotiate
 no mls qos trust dscp
 lacp port-priority 1
 channel-protocol lacp
 channel-group 16 mode active
!
interface TenGigE 8/0/17
 no shut
 description Connected to PE44_IOS-XR_Router GE 0/1/0/17
 mtu 9202
 no ip address
 logging event link-status
 logging event status
 speed nonegotiate
 no mls qos trust dscp
 lacp port-priority 2
 channel-protocol lacp
 channel-group 16 mode active
!
```

IOS side - connections to CE devices:

```
hostname P19_C7609-S

interface TenGigE 8/0/7
 description Connected to PE62_C3750-ME GE 1/0/2
 mtu 9000
 no ip address
 speed nonegotiate
 mls qos trust dscp
 service instance 160 ethernet
  description VLAN 160 over Port-Ch 16
  encapsulation dot1q 100 second-dot1q 160
  rewrite ingress tag pop 1 symmetric
 !
 service instance 161 ethernet
  description VLAN 161 over Port-Ch 16
  encapsulation dot1q 161
 !
!
connect eline-161 Port-channel16 161 TenGigE 8/0/7 161
 !
!
connect eline-160 Port-channel16 160 TenGigE 8/0/7 160
 !
!
```

IOS side - CE devices:

```
hostname PE62_C3750-ME
!
vlan 161
!
interface TenGigE 1/0/1
 description Connected to PE63_ME-C3400 GE 0/1
 switchport access vlan 100
 switchport mode dot1q-tunnel
!
interface TenGigE 1/0/2
```

```
 description Connected to P19_C7609-S GE 8/0/7
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 100,161
 switchport mode trunk
!
interface Vlan161
 description VLAN 161 over Port-Chan 16 on P19
 ip address 161.0.0.62 255.255.255.0
!
```

```
hostname PE63_ME-C3400
!
vlan 160
!
interface TenGigE 0/1
 description Connected to PE62_C3750-ME GE 1/0/1
 port-type nni
 switchport trunk allowed vlan 160
 switchport mode trunk
!
interface Vlan160
 description VLAN 160 over Port-Chan 16 on P19
 ip address 160.0.0.63 255.255.255.0
!
```

CHAPTER 5

Implementing Point to Point Layer 2 Services

This module provides conceptual and configuration information for point-to-point Layer 2 (L2) connectivity.

These point-to-point services are supported:

- Local Switching—A point-to-point circuit internal to a single Cisco ASR 9000 Series Router, also known as local connect.
- Pseudowires—A virtual point-to-point circuit from a Cisco ASR 9000 Series Router. Pseudowires are implemented over MPLS.

Note Point to Point Layer 2 Services are also called as MPLS Layer 2 VPNs.

Note For more information about Point to Point Layer 2 Services on the Cisco ASR 9000 Series Router and for descriptions of the commands listed in this module, see the "Related Documents" section.

Feature History for Implementing Point to Point Layer 2 Services

Release	Modification
Release 3.7.2	This feature was introduced.
Release 3.9.0	Scale enhancements were introduced.
Release 4.0.0	Support was added for Any Transport over MPLS (AToM) features.
Release 4.0.1	Support was added for these features: • Pseudowire Load Balancing • Any Transport over MPLS (AToM) features: • HDLC over MPLS (HDLCoMPLS) • PPP over MPLS (PPPoMPLS)

Release	Modification
Release 4.1.0	Support was added for the Flexible Router ID feature.
Release 4.2.0	Support was added for these features: • MPLS Transport Profile • Circuit EMulation (CEM) over Packet
Release 4.3.0	Support was added for the L2VPN Nonstop Routing feature.
Release 4.3.1	Support was added for these features: • L2TPv3 over IPv6 Tunnel • ATMoMPLS Cell Relay VP Mode • GTP Load Balancing
Release 5.1.0	Support was added for these features: • Two-way pseudowire (PW) for ATM/CEMoMPLS • PW grouping for Multi-Segment PW • Hot Standby PW for ATM/CEMoMPLS • MR-APS Integration with Hot-Standby PW
Release 5.1.2	Support was added for the following: • Dynamic Single Segment Pseudowire. • Faster network convergence after pseudowire failure.

Prerequisites for Implementing Point to Point Layer 2 Services

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

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

Information About Implementing Point to Point Layer 2 Services

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

Layer 2 Virtual Private Network Overview

Layer 2 Virtual Private Network (L2VPN) emulates the behavior of a LAN across an L2 switched, IP or MPLS-enabled IP network, allowing Ethernet devices to communicate with each other as they would when connected to a common LAN segment. Point-to-point L2 connections are vital when creating L2VPNs.

As Internet service providers (ISPs) look to replace their Frame Relay or Asynchronous Transfer Mode (ATM) infrastructures with an IP infrastructure, there is a need to provide standard methods of using an L2 switched, IP or MPLS-enabled IP infrastructure. These methods provide a serviceable L2 interface to customers; specifically, to provide virtual circuits between pairs of customer sites.

Building a L2VPN system requires coordination between the ISP and the customer. The ISP provides L2 connectivity; the customer builds a network using data link resources obtained from the ISP. In an L2VPN service, the ISP does not require information about a the customer's network topology, policies, routing information, point-to-point links, or network point-to-point links from other ISPs.

The ISP requires provider edge (PE) routers with these capabilities:

- Encapsulation of L2 protocol data units (PDU) into Layer 3 (L3) packets.
- Interconnection of any-to-any L2 transports.
- Emulation of L2 quality-of-service (QoS) over a packet switch network.
- Ease of configuration of the L2 service.
- Support for different types of tunneling mechanisms (MPLS, L2TPv3, IPSec, GRE, and others).
- L2VPN process databases include all information related to circuits and their connections.

Layer 2 Local Switching Overview

Local switching allows you to switch L2 data between two interfaces of the same type, (for example, Ethernet to Ethernet) and on the same router. The interfaces can be on the same line card, or on two different line cards. During these types of switching, Layer 2 address is used instead of the Layer 3 address. A local switching connection switches L2 traffic from one attachment circuit (AC) to the other. The two ports configured in a local switching connection are ACs with respect to that local connection. A local switching connection works like a bridge domain that has only two bridge ports; traffic enters one port of the local connection and leaves the other. However, because there is no bridging involved in a local connection, there is neither MAC learning nor flooding. Also, the ACs in a local connection are not in the UP state if the interface state is DOWN. (This behavior is also different when compared to that of a bridge domain.)

Local switching ACs utilize a full variety of L2 interfaces, including L2 trunk (main) interfaces, bundle interfaces, and EFPs.

Additionally, same-port local switching allows you to switch Layer 2 data between two circuits on the same interface.

ATMoMPLS with L2VPN Overview

ATMoMPLS is a type of Layer 2 point-to-point connection over an MPLS core.

To implement the ATMoMPLS feature, the Cisco ASR 9000 Series Router plays the role of provider edge (PE) router at the edge of a provider network in which customer edge (CE) devices are connected to the Cisco ASR 9000 Series Router.

Virtual Circuit Connection Verification on L2VPN

Virtual Circuit Connection Verification (VCCV) is an L2VPN Operations, Administration, and Maintenance (OAM) feature that allows network operators to run IP-based provider edge-to-provider edge (PE-to-PE) keepalive protocol across a specified pseudowire to ensure that the pseudowire data path forwarding does not contain any faults. The disposition PE receives VCCV packets on a control channel, which is associated with the specified pseudowire. The control channel type and connectivity verification type, which are used for VCCV, are negotiated when the pseudowire is established between the PEs for each direction.

Two types of packets can arrive at the disposition egress:

- Type 1—Specifies normal Ethernet-over-MPLS (EoMPLS) data packets.
- Type 2—Specifies VCCV packets.

Cisco ASR 9000 Series Router supports Label Switched Path (LSP) VCCV Type 1, which uses an inband control word if enabled during signaling. The VCCV echo reply is sent as IPv4 that is the reply mode in IPv4. The reply is forwarded as IP, MPLS, or a combination of both.

VCCV pings counters that are counted in MPLS forwarding on the egress side. However, on the ingress side, they are sourced by the route processor and do not count as MPLS forwarding counters.

Ethernet over MPLS

Ethernet-over-MPLS (EoMPLS) provides a tunneling mechanism for Ethernet traffic through an MPLS-enabled L3 core and encapsulates Ethernet protocol data units (PDUs) inside MPLS packets (using label stacking) to forward them across the MPLS network.

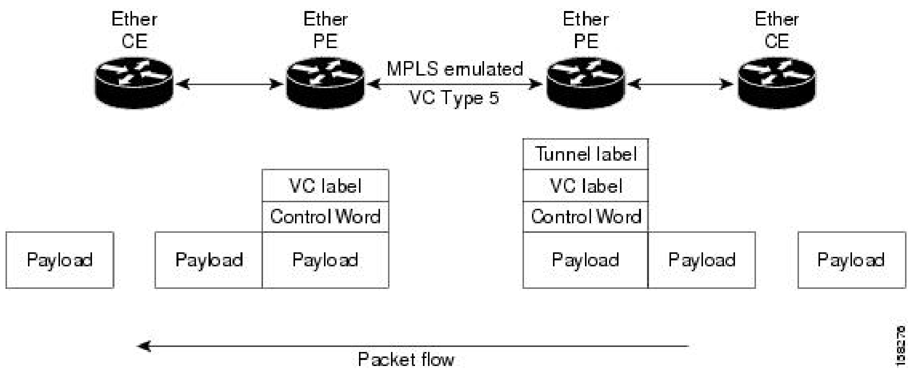
EoMPLS features are described in these subsections:

Ethernet Port Mode

In Ethernet port mode, both ends of a pseudowire are connected to Ethernet ports. In this mode, the port is tunneled over the pseudowire or, using local switching (also known as an *attachment circuit-to-attachment circuit cross-connect*) switches packets or frames from one attachment circuit (AC) to another AC attached to the same PE node.

The following figure provides an example of Ethernet port mode.

Figure 6: Ethernet Port Mode Packet Flow

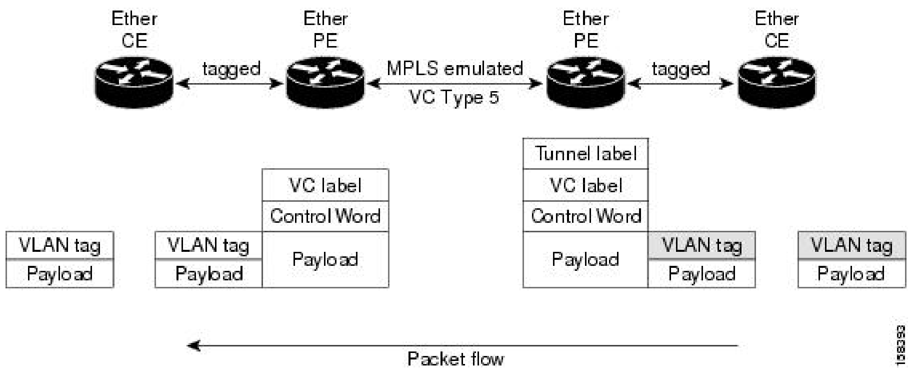


VLAN Mode

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 or VC type 5. VC type 5 is the default mode.

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 7: VLAN Mode Packet Flow



At the egress VLAN PE, the PE associates a VLAN tag to the frames coming off 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.

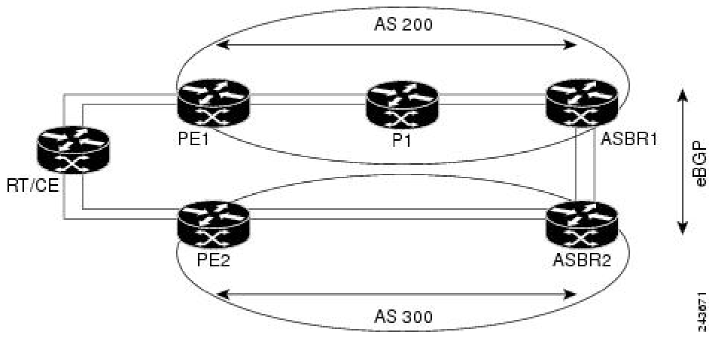
Inter-AS Mode

Inter-AS is a peer-to-peer type model that allows extension of VPNs through multiple provider or multi-domain networks. This lets service providers peer up with one another to offer end-to-end VPN connectivity over extended geographical locations.

EoMPLS support can assume 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 8: EoMPLS over Inter-AS: Basic Double AS Topology



QinQ Mode

QinQ is an extension of 802.1Q for specifying multiple 802.1Q tags (IEEE 802.1QinQ VLAN Tag stacking). Layer 3 VPN service termination and L2VPN service transport are enabled over QinQ sub-interfaces.

The Cisco ASR 9000 Series Routers implement the Layer 2 tunneling or Layer 3 forwarding depending on the subinterface configuration at provider edge routers. This function only supports up to two QinQ tags on the SPA and fixed PLIM:

- Layer 2 QinQ VLANs in L2VPN attachment circuit: QinQ L2VPN attachment circuits are configured under the Layer 2 transport subinterfaces for point-to-point EoMPLS based cross-connects using both virtual circuit type 4 and type 5 pseudowires and point-to-point local-switching-based cross-connects including full interworking support of QinQ with 802.1q VLANs and port mode.
- Layer 3 QinQ VLANs: Used as a Layer 3 termination point, both VLANs are removed at the ingress provider edge and added back at the remote provider edge as the frame is forwarded.

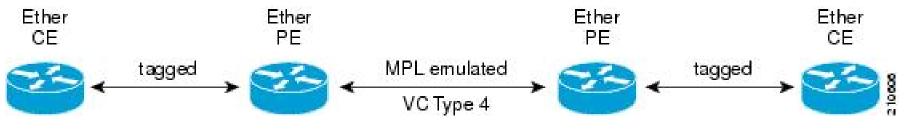
Layer 3 services over QinQ include:

- IPv4 unicast and multicast
- IPv6 unicast and multicast
- MPLS
- Connectionless Network Service (CLNS) for use by Intermediate System-to-Intermediate System (IS-IS) Protocol

In QinQ mode, each CE VLAN is carried into an SP VLAN. QinQ mode should use VC type 5, but VC type 4 is also supported. On each Ethernet PE, you must configure both the inner (CE VLAN) and outer (SP VLAN).

The following figure illustrates QinQ using VC type 4.

Figure 9: EoMPLS over QinQ Mode



QinAny Mode

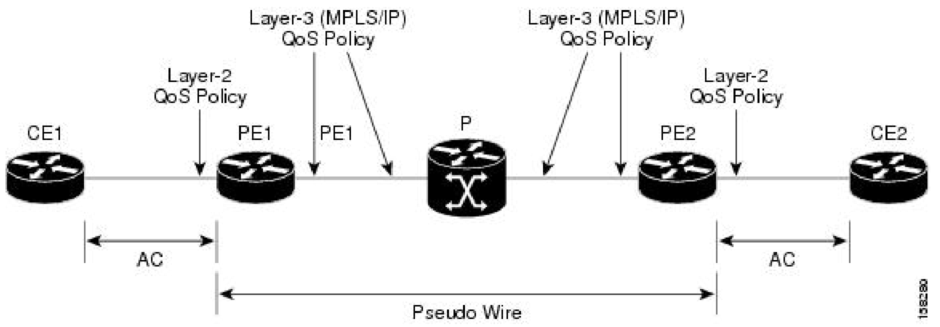
In the QinAny mode, the service provider VLAN tag is configured on both the ingress and the egress nodes of the provider edge VLAN. QinAny mode is similar to QinQ mode using a Type 5 VC, except that the customer edge VLAN tag is carried in the packet over the pseudowire, as the customer edge VLAN tag is unknown.

Quality of Service

Using L2VPN technology, you can assign a quality of service (QoS) level to both Port and VLAN modes of operation.

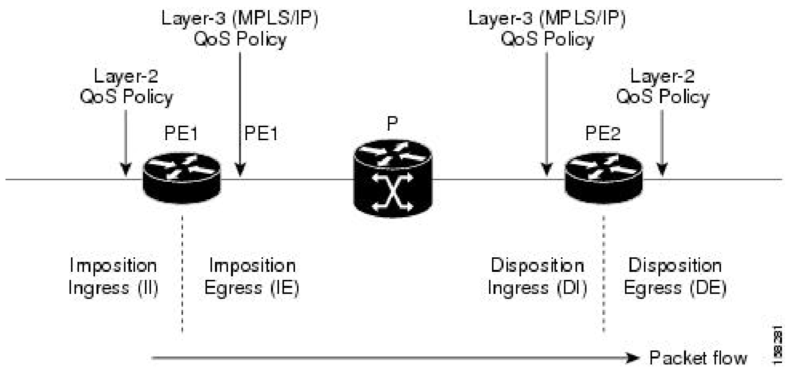
L2VPN technology requires that QoS functionality on PE routers be strictly L2-payload-based on the edge-facing interfaces (also know as *attachment circuits*). The following figure illustrates L2 and L3 QoS service policies in a typical L2VPN network.

Figure 10: L2VPN QoS Feature Application



The following figure shows four packet processing paths within a provider edge device where a QoS service policy can be attached. In an L2VPN network, packets are received and transmitted on the edge-facing interfaces as L2 packets and transported on the core-facing interfaces as MPLS (EoMPLS).

Figure 11: L2VPN QoS Reference Model



High Availability

L2VPN uses control planes in both route processors and line cards, as well as forwarding plane elements in the line cards.

The availability of L2VPN meets these requirements:

- A control plane failure in either the route processor or the line card will not affect the circuit forwarding path.
- The router processor control plane supports failover without affecting the line card control and forwarding planes.
- L2VPN integrates with existing Label Distribution Protocol (LDP) graceful restart mechanism.

Preferred Tunnel Path

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 head starts at the imposition PE router and its tail terminates on the disposition PE router).

Note

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

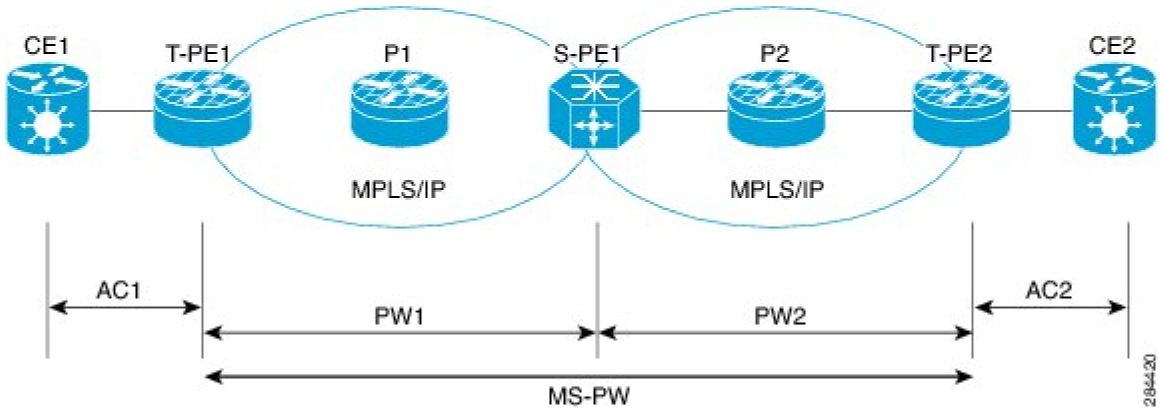
Multisegment Pseudowire

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

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

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

Figure 12: Multisegment Pseudowire: Example



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

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

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

A multisegment pseudowire is used in two general cases when:

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

  For the PW control channel to be established, the remote PE node must be accessible. Sometimes, the local PE node may not be able to access the remote node due to topology, operational, or security constraints.

  A multisegment pseudowire dynamically builds two discrete pseudowire segments and performs a pseudowire switching to establish a PW control channel between the source and destination PE nodes.

- Pseudowire Edge To Edge Emulation (PWE3) signaling and encapsulation protocols are different.

  The PE nodes are connected to networks employing different PW signaling and encapsulation protocols. Sometimes, it is not possible to use a single segment PW.

  A multisegment pseudowire, with the appropriate interworking performed at the PW switching points, enables PW connectivity between the PE nodes in the network.

Pseudowire Redundancy

Pseudowire redundancy allows you to configure your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. This feature provides the ability to recover from a failure of either the remote provider edge (PE) router or the link between the PE and customer edge (CE) routers.

L2VPNs can provide pseudowire resiliency through their routing protocols. When connectivity between end-to-end PE routers fails, an alternative path to the directed LDP session and the user data takes over. However, there are some parts of the network in which this rerouting mechanism does not protect against interruptions in service.

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

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

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

Pseudowire Load Balancing

To maximize networks while maintaining redundancy typically requires traffic load balancing over multiple links. To achieve better and more uniformed distribution, load balancing on the traffic flows that are part of the provisioned pipes is desirable. Load balancing can be flow based according to the IP addresses, Mac addresses, or a combination of those. Load balancing can be flow based according to source or destination IP addresses, or source or destination MAC addresses. Traffic falls back to default flow based MAC addresses if the IP header cannot proceed or IPv6 is be flow based.

This feature applies to pseudowires under L2VPN; this includes VPWS and VPLS.

Note Enabling virtual circuit (VC) label based load balancing for a pseudowire class overrides global flow based load balancing under L2VPN.

L2 Traffic Tunneling with IP Load Balance Hashing for MPLS Encapsulated Packets

If the frame is IP-based, the load-balancing flow “src-dst-ip” configuration causes the Layer 2 interfaces to use the IP header for flow balancing hash calculation. If the frame is not IP-based, the MAC header is used for the hash calculation. In previous releases, for an MPLS header between the MAC and IP headers, the code would use the MAC header for the flow balancing hash.

From Release 6.4.1 onwards, the code analyzes the MPLS header, and if an IP header is available, it uses that for hash calculation.

If the MPLS label stack is more than four labels deep, the code stops looking for an IP header and reverts to the MAC header hash calculation.

When L2VPN flow-based src-dst-ip load-balancing is configured, and if a payload with encapsulated GTP is used, the GTP ID is considered for load-balancing criteria. Starting from Release 6.5.1, you do not have to configure the *cef load-balancing* command to use GTP ID as a criteria for load-balancing for L2VPN scenarios. However, you must explicitly configure CEF load-balancing for Layer 3 scenarios.

Pseudowire Grouping

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

For CEM interfaces, various levels of configuration are permitted for the parent controllers, such as T1 and T3, framed or unframed. To achieve best grouping, the physical controller handle is used as the group ID.

Note Pseudowire grouping is disabled by default.

Ethernet Wire Service

An Ethernet Wire Service is a service that emulates a point-to-point Ethernet segment. This is similar to Ethernet private line (EPL), a Layer 1 point-to-point service, except the provider edge operates at Layer 2 and typically runs over a Layer 2 network. The EWS encapsulates all frames that are received on a particular UNI and transports these frames to a single-egress UNI without reference to the contents contained within the frame. The operation of this service means that an EWS can be used with VLAN-tagged frames. The VLAN tags are transparent to the EWS (bridge protocol data units [BPDUs])—with some exceptions. These exceptions include IEEE 802.1x, IEEE 802.2ad, and IEEE 802.3x, because these frames have local significance and it benefits both the customer and the Service Provider to terminate them locally.

Since the service provider simply accepts frames on an interface and transmits these without reference to the actual frame (other than verifying that the format and length are legal for the particular interface) the EWS is indifferent to VLAN tags that may be present within the customer Ethernet frames.

EWS subscribes to the concept of all-to-one bundling. That is, an EWS maps a port on one end to a point-to-point circuit and to a port on another end. EWS is a port-to-port service. Therefore, if a customer needs to connect a switch or router to n switches or routers it will need n ports and n pseudowires or logical circuits.

One important point to consider is that, although the EWS broadly emulates an Ethernet Layer 1 connection, the service is provided across a shared infrastructure, and therefore it is unlikely that the full interface bandwidth will be, or needs to be, available at all times. EWS will typically be a sub-line rate service, where many users share a circuit somewhere in their transmission path. As a result, the cost will most likely be less than that of EPL. Unlike a Layer 1 EPL, the SP will need to implement QoS and traffic engineering to meet the specific objectives of a particular contract. However, if the customer's application requires a true wire rate transparent service, then an EPL service—delivered using optical transmission devices such as DWDM (dense wavelength division multiplexing), CDWM (coarse wavelength division multiplexing), or SONET/SDH—should be considered.

IGMP Snooping

IGMP snooping provides a way to constrain multicast traffic at Layer 2. By snooping the IGMP membership reports sent by hosts in the bridge domain, the IGMP snooping application can set up Layer 2 multicast forwarding tables to deliver traffic only to ports with at least one interested member, significantly reducing the volume of multicast traffic.

Configured at Layer 3, IGMP provides a means for hosts in an IPv4 multicast network to indicate which multicast traffic they are interested in and for routers to control and limit the flow of multicast traffic in the network (at Layer 3).

IGMP snooping uses the information in IGMP membership report messages to build corresponding information in the forwarding tables to restrict IP multicast traffic at Layer 2. The forwarding table entries are in the form <Route, OIF List>, where:

- Route is a <*, G> route or <S, G> route.
- OIF List comprises all bridge ports that have sent IGMP membership reports for the specified route plus all Multicast Router (mrouter) ports in the bridge domain.

The IGMP snooping feature can provide these benefits to a multicast network:

- Basic IGMP snooping reduces bandwidth consumption by reducing multicast traffic that would otherwise flood an entire VPLS bridge domain.
- With optional configuration options, IGMP snooping can provide security between bridge domains by filtering the IGMP reports received from hosts on one bridge port and preventing leakage towards the hosts on other bridge ports.
- With optional configuration options, IGMP snooping can reduce the traffic impact on upstream IP multicast routers by suppressing IGMP membership reports (IGMPv2) or by acting as an IGMP proxy reporter (IGMPv3) to the upstream IP multicast router.

Refer to the *Implementing Layer 2 Multicast with IGMP Snooping module* in the *Cisco ASR 9000 Series Aggregation Services Router Multicast Configuration Guide* for information on configuring IGMP snooping.

The applicable IGMP snooping commands are described in the *Cisco ASR 9000 Series Aggregation Services Router Multicast Command Reference*.

IP Interworking

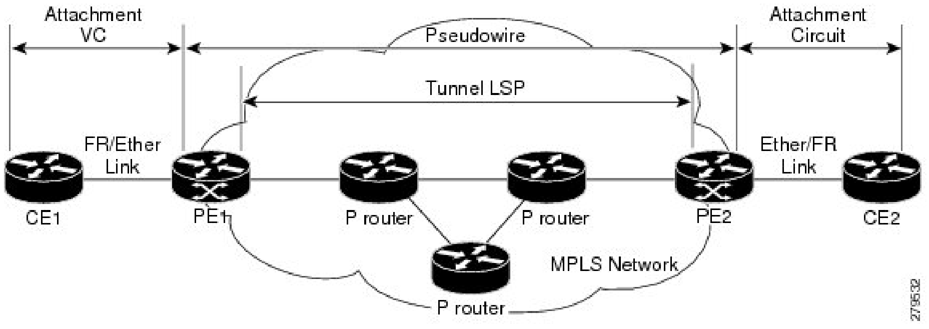
Customer deployments require a solution to support AToM with disparate transport at network ends. This solution must have the capability to translate transport on one customer edge (CE) device to another transport, for example, Frame relay to Ethernet. The Cisco ASR 9000 Series SPA Interface Processor-700 and the Cisco ASR 9000 Series Ethernet line cards enable the Cisco ASR 9000 Series Routers to support multiple legacy services.

IP Interworking is a solution for transporting Layer 2 traffic over an IP/MPLS backbone. It accommodates many types of Layer 2 frames such as Ethernet and Frame Relay using AToM tunnels. It encapsulates packets at the provider edge (PE) router, transports them over the backbone to the PE router on the other side of the cloud, removes the encapsulation, and transports them to the destination. The transport layer can be Ethernet on one end and Frame relay on the other end. IP interworking occurs between disparate endpoints of the AToM tunnels.

Note Only routed interworking is supported between Ethernet and Frame Relay based networks for MPLS and Local-connect scenarios.

The following figure shows the interoperability between an Ethernet attachment VC and a Frame Relay attachment VC.

Figure 13: IP Interworking over MPLS Core



An attachment circuit (AC) is a physical or logical port or circuit that connects a CE device to a PE device. A pseudowire (PW) is a bidirectional virtual connection (VC) connecting two ACs. In an MPLS network, PWs are carried inside an LSP tunnel. The core facing line card on the PE1 and PE2 could be a Cisco ASR 9000 Series SPA Interface Processor-700 or a Cisco ASR 9000 Series Ethernet line card.

In the IP Interworking mode, the Layer 2 (L2) header is removed from the packets received on an ingress PE, and only the IP payload is transmitted to the egress PE. On the egress PE, an L2 header is appended before the packet is transmitted out of the egress port.

In Figure above , CE1 and CE2 could be a Frame Relay (FR) interface or a GigabitEthernet (GigE) interface. Assuming CE1 is a FR and CE2 is either a GigE or dot1q, or QinQ. For packets arriving from an Ethernet CE (CE2), ingress LC on the PE (PE2) facing the CE removes L2 framing and forwards the packet to egress PE (PE1) using IPoMPLS encapsulation over a pseudowire. The core facing line card on egress PE removes the MPLS labels but preserves the control word and transmits it to the egress line card facing FR CE (CE1). At the FR PE, after label disposition, the Layer 3 (L3) packets are encapsulated over FR.

Similarly, IP packets arriving from the FR CE are translated into IPoMPLS encapsulation over the pseudowire. At the Ethernet PE side, after label disposition, the PE adds L2 Ethernet packet header back to the packet before transmitting it to the CE, as the packets coming out from the core carry only the IP payload.

These modes support IP Interworking on AToM:

- Ethernet to Frame Relay

  Packets arriving from the Ethernet CE device have MAC (port-mode, untagged, single, double tag), IPv4 header and data. The Ethernet line card removes the L2 framing and then forwards the L3 packet to the egress line card. The egress line card adds the FR L2 header before transmitting it from the egress port.

- Ethernet to Ethernet

  Both the CE devices are Ethernet. Each ethernet interface can be port-mode, untagged, single, or double tag, although this is not a typical scenario for IP interworking.

AToM iMSG

This feature enables an interworking layer in the access network(s) to terminate all non-Ethernet functionality and translate these connections to a Ethernet centric service which can be terminated on the Layer 3 edge routers. Currently, the time-division multiplexing (TDM) based services terminate on the Layer 3 edge routers directly. A simplified and more cost optimized model for the L3 networks is enabled by moving the TDM complexity into the access layer.

The Layer 2 encapsulation is removed from an IP packet by the ingress PE's attachment circuit facing ingress line card. The MPLS encapsulated IP packet payload is then sent across the fabric to the core facing egress line card. The egress line card then transmits the packet through the MPLS core. On the remote PE, the MPLS label is removed, Layer 2 header of the egress AC is added and finally the packet is sent to the connected CE. L2VPN VPWS has been enhanced to support:

- Point-to-Point Protocol (PPP)
- High-level Data Link Control (HDLC)
- Multilink Point-to-Point Protocol (MLPPP)
- QOS support for all the encapsulation types

  For more information on QoS, see the *Cisco ASR 9000 Series Aggregation Services Router Modular Quality of Service Configuration*.

The TDM ACs can be configured on these SPAs:

- SPA-8XCHT1/E1
- SPA-4XCT3/DS0
- SPA-1XCHSTM1/OC3
- SPA-2XCHOC12/DS0
- SPA-1XCHOC48/DS3
- SPA-4XT3/E3
- SPA-4XOC3-POS-V2
- SPA-8XOC3-POS
- SPA-8XOC12-POS
- SPA-1XOC48POS/RPR
- SPA-2XOC48POS/RPR

Any Transport over MPLS

Any Transport over MPLS (AToM) transports Layer 2 packets over a Multiprotocol Label Switching (MPLS) backbone. This enables service providers to connect customer sites with existing Layer 2 networks by using a single, integrated, packet-based network infrastructure. Using this feature, service providers can deliver Layer 2 connections over an MPLS backbone, instead of using separate networks.

AToM encapsulates Layer 2 frames at the ingress PE router, and sends them to a corresponding PE router at the other end of a pseudowire, which is a connection between the two PE routers. The egress PE removes the encapsulation and sends out the Layer 2 frame.

The successful transmission of the Layer 2 frames between PE routers is due to the configuration of the PE routers. You set up a connection, called a *pseudowire*, between the routers. You specify this information on each PE router:

- The type of Layer 2 data that will be transported across the pseudowire, such as Ethernet and Frame Relay
- The IP address of the loopback interface of the peer PE router, which enables the PE routers to communicate.
- A unique combination of peer PE IP address and VC ID that identifies the pseudowire.

Control Word Processing

The control word contains forward explicit congestion notification (FECN), backward explicit congestion notification (BECN) and DE bits in case of frame relay connection.

Control word is mandatory for:

- Frame Relay
- ATM AAL5
- Frame Relay to Ethernet bridged interworking
- cHDLC/PPP IP interworking
- CEM (Circuit Emulation)

The system does not map bits from one transport end point to another across an AToM IP Interworking connection.

Whenever supported, control word is also recommended for pseudowires, as it enables proper load balancing without packet desequencing independent of L2VPN packet content. Without control word the heuristics used to perform load balancing cannot achieve optimal results in all cases.

High-level Data Link Control over MPLS

The attachment circuit (AC) is a main interface configured with HDLC encapsulation. Packets to or from the AC are transported using an AToM pseudowire (PW) of VC type 0x6 to or from the other provider edge (PE) router over the MPLS core network.

With HDLC over MPLS, the entire HDLC packet is transported. The ingress PE router removes only the HDLC flags and FCS bits.

PPP over MPLS

The attachment circuit (AC) is a main interface configured with PPP encapsulation. Packets to or from the AC are transported through an AToM PW of VC type 0x7 to or from the other provider edge (PE) routers over the MPLS core network.

With PPP over MPLS, the ingress PE router removes the flags, address, control field, and the FCS bits.

Frame Relay over MPLS

Frame Relay over MPLS (FRoMPLS) provides leased line type of connectivity between two Frame Relay islands. Frame Relay traffic is transported over the MPLS network.

Note The Data Link Connection Identifier (DLCI) DCLI-DLCI mode is supported. A control word (required for DLCI-DLCI mode) is used to carry additional control information.

When a Provider Edge (PE) router receives a Frame Relay protocol packet from a subscriber site, it removes the Frame Relay header and Frame Check Sequence (FCS) and appends the appropriate Virtual Circuit (VC) label. The removed Backward Explicit Congestion Notification (BECN), Forward Explicit Congestion Notification (FECN), Discard Eligible (DE) and Command/Response (C/R) bits are (for DLCI-DLCI mode) sent separately using a control word.

MPLS Transport Profile

MPLS transport profile (MPLS-TP) tunnels provide the transport network service layer over which IP and MPLS traffic traverse. Within the MPLS-TP environment, pseudowires (PWs) use MPLS-TP tunnels as the transport mechanism. MPLS-TP tunnels help transition from SONET/SDH TDM technologies to packet switching, to support services with high bandwidth utilization and low cost. Transport networks are connection oriented, statically provisioned, and have long-lived connections. Transport networks usually avoid control protocols that change identifiers (like labels). MPLS-TP tunnels provide this functionality through statically provisioned bidirectional label switched paths (LSPs).

For more information on configuring MPLS transport profile, refer to the *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide*.

MPLS-TP supports these combinations of static and dynamic multisegment pseudowires:

- Static-static
- Static-dynamic
- Dynamic-static
- Dynamic-dynamic

MPLS-TP supports one-to-one L2VPN pseudowire redundancy for these combinations of static and dynamic pseudowires:

- Static pseudowire with a static backup pseudowire
- Static pseudowire with a dynamic backup pseudowire
- Dynamic pseudowire with a static backup pseudowire
- Dynamic pseudowire with a dynamic backup pseudowire

The existing TE preferred path feature is used to pin down a PW to an MPLS-TP transport tunnel. See Preferred Tunnel Path for more information on configuring preferred tunnel path. For a dynamic pseudowire, PW status is exchanged through LDP whereas for static PW, status is transported in PW OAM message. See Configuring PW Status OAM for more information on configuring PW status OAM. By default, alarms are not generated when the state of a PW changes due to change in the state of MPLS TP tunnel carrying that PW.

Circuit Emulation Over Packet Switched Network

Circuit Emulation over Packet (CEoP) is a method of carrying TDM circuits over packet switched network. CEoP is similar to a physical connection. The goal of CEoP is to replace leased lines and legacy TDM networks.

CEoP operates in two major modes:

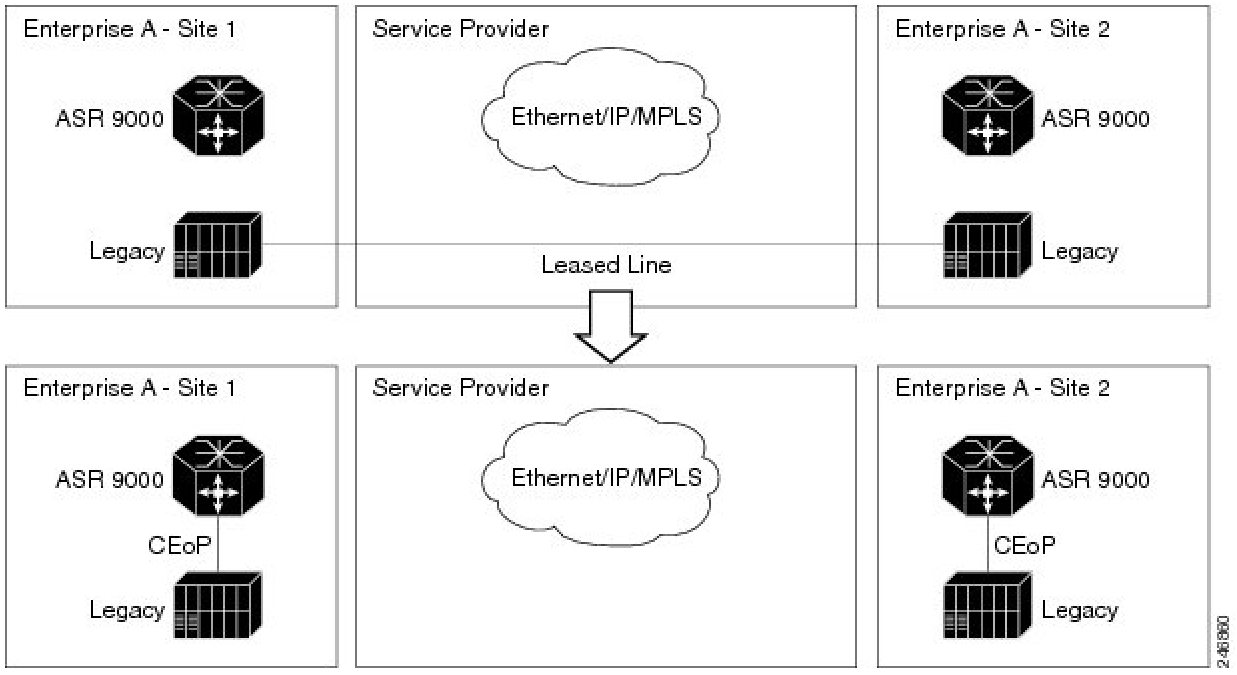
- Unstructured mode is called SAToP (Structure Agnostic TDM over Packet)

  SAToP addresses only structure-agnostic transport, i.e., unframed E1, T1, E3 and T3. It segments all TDM services as bit streams and then encapsulates them for transmission over a PW tunnel. This protocol can transparently transmit TDM traffic data and synchronous timing information. SAToP completely disregards any structure and provider edge routers (PEs) do not need to interpret the TDM data or to participate in the TDM signaling. The protocol is a simple way for transparent transmission of PDH bit-streams.

- Structured mode is named CESoPSN (Circuit Emulation Service over Packet Switched Network)

  Compared with SAToP, CESoPSN transmits emulated structured TDM signals. That is, it can identify and process the frame structure and transmit signaling in TDM frames. It may not transmit idle timeslot channels, but only extracts useful timeslots of CE devices from the E1 traffic stream and then encapsulates them into PW packets for transmission.CEoP SPAs are half-height (HH) Shared Port Adapters (SPA) and the CEoP SPA family consists of 24xT1/E1, 2xT3/E3, and 1xOC3/STM1 unstructured and structured (NxDS0) quarter rate, half height SPAs.

Figure 14: Enterprise Data Convergence using Circuit Emulation over Packet



The CEM functionality is supported only on Engine 5 line cards having CEoP SPAs. CEM is supported on:

- 1-port Channelized OC3 STM1 ATM CEoP SPA (SPA-1CHOC3-CE-ATM)

  CESoPSN and SAToP can use MPLS, UDP/IP, and L2TPv3 as the underlying transport mechanism. This release supports only MPLS transport mechanism.

CEoP SPA supports these modes of operation:

- Circuit Emulation Mode (CEM)
- ATM Mode
- IMA Mode

Note Only CEM mode is supported.

Benefits of Circuit Emulation over Packet Switched Network

CEM offers these benefits to the service provider and end-users:

- Saving cost in installing equipment.
- Saving cost in network operations; as leased lines are expensive, limiting their usage to access only mode saves significant costs.
- Ensuring low maintenance cost because only the core network needs to be maintained.
- Utilizing the core network resources more efficiently with packet switched network, while keeping investment in access network intact.
- Providing cheaper services to the end-user.

L2VPN Nonstop Routing

The L2VPN Nonstop Routing (NSR) feature avoids label distribution path (LDP) sessions from flapping on events such as process failures (crash) and route processor failover (RP FO). NSR on process failure (crash) is supported by performing RP FO, if you have enabled NSR using NSR process failure switchover.

NSR enables the router (where failure has occurred) to maintain the control plane states without a graceful restart (GR). NSR, by definition, does not require any protocol extension and typically uses Stateful Switch Over (SSO) to maintain it's control plane states.

Note NSR is enabled by default for L2VPN on Cisco IOS XR 64 bit operating system. You cannot configure the **nsr** command under L2VPN configuration submode.

L2TPv3 over IPv6

A L2TPv3 over IPv6 tunnel is a static L2VPN cross-connect that uses Layer 2 Tunneling Protocol version 3 (L2TPv3) over IPv6, with a unique IPv6 source address for each cross-connect. The L2TPv3 over IPv6 tunnels consists of one L2TPv3 tunnel for each subscriber VLAN. The unique IPv6 address completely identifies the customer, and the service that is delivered.

Note L2TPv3 over IPv6 tunnels are supported on the ASR 9000 2nd and 3rd Generation line cards by a scale of 15000 crossconnects for each router and line card.

Note L2TPv3 over IPv6 tunnels are not supported on the ASR 9000 4th and 5th Generation line cards.

Note nV satellite access interfaces do not support L2TPv3 over IPv6.

Overview

L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network using Layer 2 virtual private networks (VPNs). Traffic between two customer network sites is encapsulated in IP packets carrying L2TP data messages (payload) and sent across an IP network. The backbone routers of the IP network treat this payload in the same manner as it treats any other IP traffic. Implementing L2TPv3 over IPv6 provides an opportunity to utilize unique source IPv6 addresses to directly identify Ethernet attachment circuits. In this case, processing of the L2TPv3 session ID is bypassed; this is because each tunnel has only one associated session. This local optimization, however, does not hinder the ability to continue supporting circuit multiplexing through the session ID for other L2TPv3 tunnels on the same router.

For more information on:

- Configuration procedures, see Configuring L2TPv3 over IPv6 Tunnels.
- Configuration examples, see Configuring L2TPv3 over IPv6 Tunnels: Example.

Traffic Injection from L2TPv3 over IPv6 Tunnel

Traffic Injection from L2TPv3 over IPv6 Tunnel feature allows you to inject diagnostic traffic through Layer 2 Tunneling Protocol version 3 (L2TPv3) Switched Port Analyzer (SPAN) tunnel. The diagnostic traffic allows you to monitor and troubleshoot the network traffic. You can send the diagnostic traffic from customer office (CO) using traffic generator towards the customer or towards the network.

In previous releases, with traffic mirroring feature the user was only able to send the mirrored traffic from customer towards the monitoring device, the mirror tunnel was unidirectional.

Traffic mirroring copies traffic from one or more source ports and sends the copied traffic to one or more destinations for analysis by a network analyzer or other monitoring device. Traffic mirroring does not affect the flow of traffic on the source interfaces or sub-interfaces, and allows the mirrored traffic to be sent to a destination interface or sub-interface

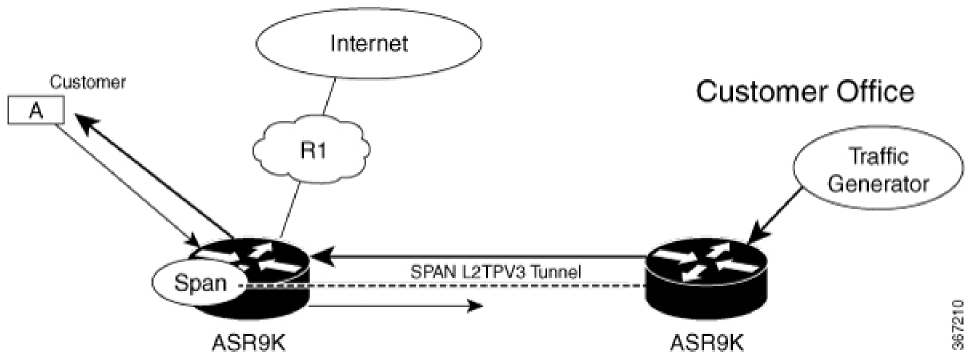
Restrictions

- This feature is not supported on bundle and sub-bundle interfaces, supported only on main and sub interfaces.

- Diagnostic traffic directed from network to customer does not traverse the same path as the actual non-diagnostic network to customer traffic. As a result, any issue with the core facing interface is not diagnosed.
- Diagnostic traffic will mix with the actual customer traffic. It is the responsibility of CO to ensure that it does not cause any problems to the customer, and CO is able to differentiate diagnostic traffic from customer traffic in the SPAN tunnel.
- Physical port features are not available in the inward inject path, so the problems in the physical port features, such as, EOAM or BIA MAC are not diagnosed.
- Bundle and other hash calculations, such as, ECMP, are not available, so the only customer interface supported is a physical interface.
- For Layer 3, only IPv4 and IPv6 diagnostic payload is supported.

Topology

Figure 15: Traffic Injection from L2TPv3 over IPv6 Tunnel



Consider a topology where an L2TPv3 tunnel is created between two ASR 9000 devices. Customer Office (CO) sends diagnostic traffic using traffic generator over L2TPv3 over IPv6 tunnel to the ASR 9000 router. The ASR 9000 router on the left-hand side sends the diagnostic traffic towards the customer as though it is sent from the network or sends the diagnostic traffic towards the network as though it is sent from the customer.

The diagnostic traffic header contains destination MAC address, source MAC address, and IP payload. If the header contains destination MAC address of the ASR 9000 router, the diagnostic traffic is sent to the network as though it sent from the customer. If the header contains source MAC address of the ASR 9000 router, the diagnostic traffic is sent to the customer as though it sent from the network.

Configure Traffic Injection from L2TPv3 over IPv6 Tunnel

Perform these tasks on ASR 9000 router, which is on the left-hand side to configure Traffic Injection from L2TPv3 over IPv6 Tunnel feature,

- Create a pseudowire monitor session with inject interface
- Attach the monitor session to an interface which needs to be spanned
- Configure L2VPN xconnect with monitor session

```
/* Create a pseudowire monitor session with inject interface */
```

```
Router# configure
Router(config)# monitor-session span1
Router(config-mon)# destination pseudowire
Router(config-mon)# inject-interface tenGigE 0/1/0/0/0
Router(config-mon)# commit
Router(config-mon)# end

/* Attach the monitor session to an interface which needs to be spanned */

Router# configure
Router(config)# int tenGigE 0/1/0/0/0
Router(config-subif)# monitor-session span1 ethernet
Router(config-if-mon)# commit

/* Configure L2VPN  xconnect with monitor session */

Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xc-span1
Router(config-l2vpn-xc)# p2p span-session1
Router(config-l2vpn-xc-p2p)# monitor-session span1
Router(config-l2vpn-xc-p2p)# neighbor ipv6 1112::1:1 pw-id 101
Router(config-l2vpn-xc-p2p-pw)# pw-class ts
Router(config-l2vpn-xc-p2p-pw)# source 1111::1:1
Router(config-l2vpn-xc-p2p-pw)# l2tp static local cookie size 8 value 0x1 0xa1 local session
 101
Router(config-l2vpn-xc-p2p-pw)# l2tp static remote cookie size 8 value 0xa1 0x1 remote
session 101
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# end
```

Running Configuration

```
configure
 monitor-session span1
  destination pseudowire
  inject-interface tenGigE 0/1/0/0/0
  !
!

configure
 int tenGigE 0/1/0/0/0
  monitor-session span1 ethernet
  !
!

l2vpn
 xconnect group xc-span1
  p2p span-session1
   monitor-session span1
   neighbor ipv6 1112::1:1 pw-id 101
    pw-class ts
    source 1111::1:1
    l2tp static
     local cookie size 8 value 0x1 0xa1
     local session 101
     remote cookie size 8 value 0xa1 0x1
     remote session 101
    !
   !
  !
```

Verification

Verify that the source MAC address and destination MAC address are matching.

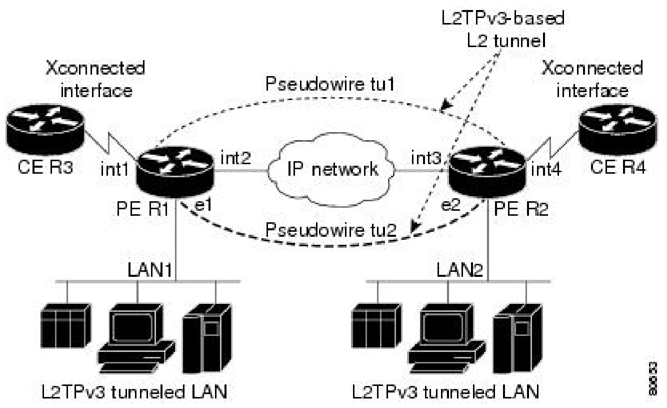
```
/* Verify that the source MAC address is matching */

Router#show monitor-session span1 counters
Monitor-session span1
  TenGigE0/1/0/0/0.1
    Rx replicated: 248 packets, 247086 octets
    Tx replicated: 20001 packets, 20000094 octets
    Non-replicated: 0 packets, 0 octets


Router#show interface tenGigE 0/1/0/7/8 accounting
TenGigE0/1/0/7/8.1
  Protocol              Pkts In         Chars In     Pkts Out        Chars Out
  IPV6_UNICAST            10001         10480072        10005         10480632
  IPV6_MULTICAST              1              104            0                0
  IPV6_ND                     2              212            1               72


Router#show interface tenGigE 0/1/0/0/0 accounting
TenGigE0/1/0/0/0.1
  Protocol              Pkts In         Chars In     Pkts Out        Chars Out
  IPV6_UNICAST                2              136        10002          9780144
  IPV6_ND

/* Verify that the destination MAC address is matching */

Router#show monitor-session counters
Monitor-session span1
  TenGigE0/1/0/0/0.1
    Rx replicated: 10001 packets, 10000094 octets
    Tx replicated: 1 packets, 94 octets
    Non-replicated: 0 packets, 0 octets


Router#show interface tenGigE 0/1/0/7/8 accounting
TenGigE0/1/0/7/8.1
  Protocol              Pkts In         Chars In     Pkts Out        Chars Out
  IPV6_UNICAST            10000         10480000        10000         10480000
  IPV6_MULTICAST              0                0            1              104
  IPV6_ND                     0                0            1              104


Router#show interface tenGigE 0/1/0/0/0 accounting
TenGigE0/1/0/0/0.1
  Protocol              Pkts In         Chars In     Pkts Out        Chars Out
  IPV6_UNICAST            10000          9780000            0                0
  IPV6_ND                     1               82            1               72
```

L2TPv3 over IPv4

Layer 2 Tunneling Protocol version 3 (L2TPv3) over IPv4 provides a dynamic mechanism for tunneling Layer 2 (L2) circuits across a packet-oriented data network, with multiple attachment circuits multiplexed over a single pair of IP address endpoints, using the L2TPv3 session ID as a circuit discriminator.

The following figure shows how the L2TPv3 feature is used to set up VPNs using Layer 2 tunneling over an IP network. All traffic between two customer network sites is encapsulated in IP packets carrying L2TP data messages and sent across an IP network. The backbone routers of the IP network treat the traffic as any other IP traffic and needn't know anything about the customer networks.

Figure 16: L2TPv3 Operation



In the above figure the PE routers R1 and R2 provide L2TPv3 services. The R1 and R2 routers communicate with each other using a pseudowire over the IP backbone network through a path comprising the interfaces *int1* and *int2*, the IP network, and interfaces *int3* and *int4*. The CE routers R3 and R4 communicate through a pair of cross-connected Ethernet or 802.1q VLAN interfaces using an L2TPv3 session. The L2TPv3 session tu1 is a pseudowire configured between interface int1 on R1 and interface int4 on R2. Any packet arriving on interface int1 on R1 is encapsulated and sent through the pseudowire control-channel (tu1) to R2. R2 decapsulates the packet and sends it on interface int4 to R4. When R4 needs to send a packet to R3, the packet follows the same path in reverse.

Note L2TPv3 over IPv4 tunnels are supported on only the ASR 9000 2nd and 3rd Generation line cards and not supported on the ASR 9000 4th and 5th Generation line cards.

Note nV satellite access interfaces do not support L2TPv3 over IPv4.

For more information on:

- Configuration procedures, see Configuring L2TPv3 over IPv4 Tunnels, on page 174.
- Configuration examples, see Configuring L2TPv3 over IPv4 Tunnels: Example, on page 205.

Dynamic Single-Segment Pseudowire

A single-segment pseudowire (SS-PW) is a point-to-point pseudowire (PW) where the PW segment is present between two PE routers.

In this feature, a single-segment pseudowire is established between two PE routers of the same autonomous system (AS) dynamically using the FEC 129 information. The objective of this feature is to ensure interoperability of the Cisco routers with the third-party routers.

Active and Passive Signaling

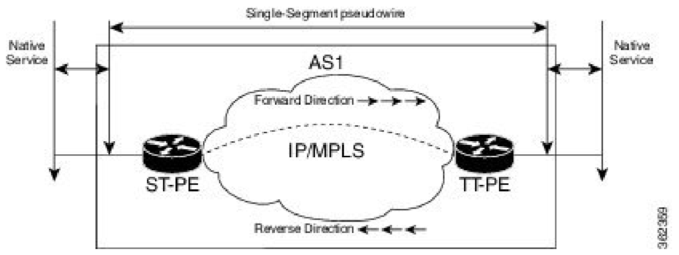
The T-PE on which the SS-PW is initiated and the signaling message is transmitted from is called as the source-terminating PE (ST-PE). The T-PE that waits and responds to the SS-PW signaling message is called the target-terminating PE (TT-PE).

The signaling flow from the ST-PE to TT-PE is referred to as the forward direction signaling or the active signaling. The signaling flow from the TT-PE to ST-PE is referred to as the reverse direction signaling or the passive signaling.

Generally, the PE with the highest prefix address takes the active role and becomes the ST-PE, and the other PE becomes the passive TT-PE.

The following figure illustrates the SS-PW signaling flow between ST-PE and TT-PE:

Figure 17: Single-Segment Pseudowire Between ST-PE and TT-PE



Functionality of Dynamic Single Segment Pseudowire

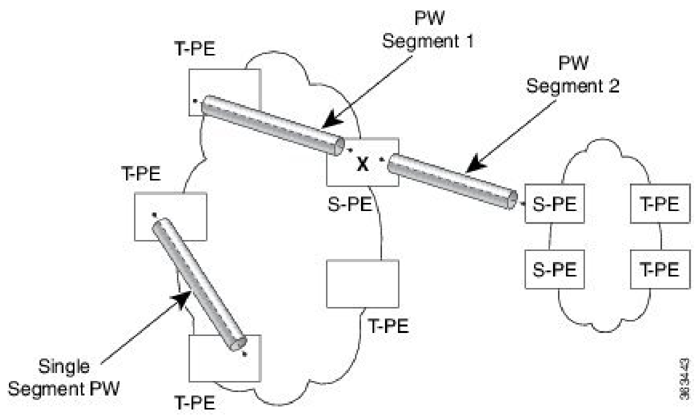
The dynamic discovery of the pseudowire path from the ST-PE to the T-PE is achieved using the L2 route table. The route table entries, that is, a list of prefix and associated next-hops to the L2VPN are populated by BGP.

Note In Release 5.1.2, Cisco supports only the routable prefix to reach the TAII on the T-PE. The routable prefix is the neighbor address of the targeted-LDP session. The reachability of packets from the source to the destination is achieved by user configurations (see Configuring L2VPN Single Segment Pseudowire, on page 99) However, BGP supports MS-PW Subsequent Address Family Identifiers (SAFI) that is used to exchange the L2 routes across all the PEs. SS-PW uses the BGP MS-PW address family to function. To ensure interoperability with other third-party routers, Cisco advertises a single BGP MS-PW route per T-PE where the value of AC-ID (attachment circuit-identifier) is a wild-card entry.

The supported pseudowire features are pw-status, pw-grouping, and tag-impose vlan.

The following figure illustrates the E-line Services Network with SS-PWs:

Figure 18: E-Line Services Network with SS-PWs



Prerequisites for Configuring L2VPN Single Segment Pseudowires

MPLS LDP, IGP, BGP, L2VPN, and interfaces must be configured on the two end points of the PW:

- Configuring MPLS Label Distribution Protocol.
- Configuring Interior Gateway Protocol (IGP).
- Configure Border Gateway Protocol (BGP).
- Configuring an Interface or Connection for L2VPN.

Restrictions for Configuring L2VPN Single Segment Pseudowires

- The routed pseudowire can only be enabled on Virtual Private Wire Service (VPWS) cross connects.
- A cross-connect cannot have both ends configured as “neighbor routed” pseudowire.
- SS-PW is not supported as the other member of the cross-connect, that is, at a T-PE, one end of the cross-connect can be the termination of the SS-PW and the other end can either be an attachment circuit (AC) or a PW-HE.
- Source AII and AC-ID (attachment circuit identifier) are unique per router.
- L2TP and MPLS static are not supported.

Configuring L2VPN Single Segment Pseudowire

To configure single segment pseudowire in the network, do the following:

1. (Optional) Configuring the related L2VPN Global Parameters. See Configuring L2VPN Global Parameters

   This procedure is used to overwrite the default BGP Route Distinguisher (RD) auto-generated value and also the Autonomous System Number (ASN) and Route Identifier (RID) of BGP.

2. Configuring L2VPN VPWS SS-PW

3. Configuring L2VPN MS-PW Address Family Under BGP

   The address family is configured under BGP to exchange the dynamic pseudowire routes.

Configuring L2VPN Global Parameters

Perform this task to configure L2VPN global parameters.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **router-id** *router-id*
4. **pw-routing**
5. **global-id** *global-id*
6. **bgp**
7. **rd** *route-distinguisher*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 3 **router-id** *router-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router 2.2.2.2
```

Specifies the router ID.

Step 4 **pw-routing**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-routing
```

Enables pseudowire routing capabilities and enters the pseudowire routing configuration submode

Step 5 **global-id** *global-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwr)# global-id 1000
```

Configures the L2VPN global ID value for the router.

Step 6 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwr)# bgp
```

Enables the BGP pseudowire routing capabilities and enters the BGP configuration submode.

Step 7 **rd** *route-distinguisher*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwr-bgp)# rd 192.168.1.3:10
```

Configures the BGP route distinguisher.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2VPN VPWS SS-PW

Perform this task to configure L2VPN VPWS SS-PWs.

SUMMARY STEPS

1. **configure**
2. **interface type***interface-path-id*
3. **l2vpn**
4. **xconnect group** *group-name*
5. **p2p** *xconnect-name*
6. **interface** *type interface-path-id*
7. **neighbor routed** *global-id: prefix: ac-id* **source** *ac-id*
8. (optional) **pw-class** *class-name*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:routerRP/0/RP00RSP0/CPU0:router# interface TenGigE0/1/0/12
```

Enters interface configuration mode and configures an interface.

Step 3 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 4 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group pw-he1
```

Configures a cross-connect group name using a free-format 32-character string.

Step 5 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p pw-ss
```

Enters P2P configuration submode.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/9
```

Specifies the interface type and instance.

Step 7 **neighbor routed** *global-id: prefix: ac-id* **source** *ac-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor routed 100:2.2.2.2:10 source 10
```

Enables pseudowire routing configuration submode for the p2p cross-connect.

Step 8 (optional) **pw-class** *class-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pwr)# pw-class dynamic_sspw
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring L2VPN MS-PW Address Family Under BGP

Perform this task to configure L2VPN MS-PW address family under BGP:

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family l2vpn mspw**
4. **neighbor** *ip-address*
5. **address-family l2vpn mspw**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```

Enters router configuration mode for the specified routing process.

Step 3 **address-family l2vpn mspw**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn mspw
```

Specifies the L2VPN address family and enters address family configuration mode.

Step 4 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.1
```

Adds the IP address of the neighbor in the specified autonomous system.

Step 5 **address-family l2vpn mspw**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# address-family l2vpn mspw
```

Specifies the L2VPN address family of the neighbor and enters address family configuration mode.

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

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

end - Prompts user to take one of these actions:

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

EVPN Virtual Private Wire Service (VPWS)

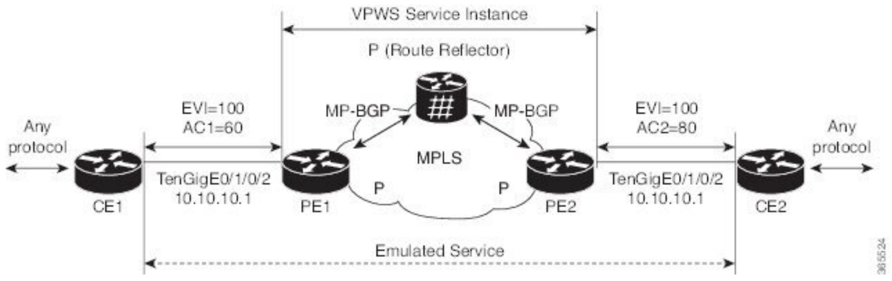
The EVPN-VPWS is a BGP control plane solution for point-to-point services. It implements the signaling and encapsulation techniques for establishing an EVPN instance between a pair of PEs. It has the ability to forward traffic from one network to another without MAC lookup. The use of EVPN for VPWS eliminates the need for signaling single-segment and multi-segment PWs for point-to-point Ethernet services. You can also configure the PWHE interface and a bridge domain access pseudowire using EVPN-VPWS.

EVPN-VPWS single homed technology works on IP and MPLS core; IP core to support BGP and MPLS core for switching packets between the endpoints.

Information About EVPN-VPWS Single Homed

The EVPN-VPWS single homed solution requires per EVI Ethernet Auto Discovery route. EVPN defines a new BGP Network Layer Reachability Information (NLRI) used to carry all EVPN routes. BGP Capabilities Advertisement used to ensure that two speakers support EVPN NLRI (AFI 25, SAFI 70) as per RFC 4760.

The architecture for EVPN VPWS is that the PEs run Multi-Protocol BGP in control-plane. The following image describes the EVPN-VPWS configuration:



- The VPWS service on PE1 requires the following three elements to be specified at configuration time:
  - The VPN ID (EVI)
  - The local AC identifier (AC1) that identifies the local end of the emulated service.
  - The remote AC identifier (AC2) that identifies the remote end of the emulated service.

  PE1 allocates a MPLS label per local AC for reachability.
- The VPWS service on PE2 is set in the same manner as PE1. The three same elements are required and the service configuration must be symmetric.

  PE2 allocates a MPLS label per local AC for reachability.
- PE1 advertise a single EVPN per EVI Ethernet AD route for each local endpoint (AC) to remote PEs with the associated MPLS label.

  PE2 performs the same task.
- On reception of EVPN per EVI EAD route from PE2, PE1 adds the entry to its local L2 RIB. PE1 knows the path list to reach AC2, for example, next hop is PE2 IP address and MPLS label for AC2.

  PE2 performs the same task.

Benefits of EVPN-VPWS

The following are the benefits of EVPN-VPWS:

- Scalability is achieved without signaling pseudowires.
- Ease of provisioning
- Pseudowires (PWs) are not used.
- Leverages BGP best path selection (optimal forwarding).

Prerequisites for EVPN-VPWS

- Ensure BGP is configured for EVPN SAFI.
- BGP session between PEs with 'address-family l2vpn evpn' to exchange EVPN routes.

Restrictions for EVPN-VPWS

- The VPN ID is unique per router.
- When specifying a list of route targets, they must be unique per PE (per BGP address-family).

- On versions earlier than IOS XR release 7.0.x, MTU is not signaled and the MTU mismatch is ignored with no interoperability issues.

  On versions later than IOS XR release 7.0.x, L3 MTU is advertised by default and the MTU mismatch is enforced by default. But this results in interoperability issues with IOS XR release 7.3.2, if transmit-l2-mtu is configured since L3 and L2 MTUs do not match. You can configure **transmit-mtu-zero** and **ignore-mtu-mismatch** commands to avoid this situation.

  On versions later than IOS XR release 7.3.2, MTU of 0 is advertised by default, and the MTU mismatch is ignored by default. L2 MTU can be advertised using the **transmit-l2-mtu** command, and MTU mismatch can be enforced with **enforce-mtu-mismatch** command.

EVPN Head End Multi-Homed

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Head End Multi-Homed	Release 7.3.1	To backhaul Layer 3 services on service PE devices over Layer 2 networks, a resilient Layer 2 service for indirectly connected end users and protection against Layer 3 service failures is necessary. This feature enables multihoming by providing redundant network connectivity-allowing you to connect a customer site to multiple PE devices. When a failure is detected, the redundant PE routers provide network service to the customer site. This feature also enables configuring pseudowire (PW-ether) that acts as a backup connection between PE routers and customer devices, thus maintaining Layer 2 services in the event of failures. The **PW-Ether** keyword is added to the interface (EVPN) command.

Increasingly, your customers are looking for efficient methods to backhaul Layer 3 services on their service PE devices over Layer 2 networks, while still being able to monitor and provide assurances on per-service granularity. To achieve this efficiency, a key requirement is resilient Layer 3 service for their non-directly connected end users and to be able to protect against active Layer 3 service failures. In achieving Layer 3 gateway and service redundancy, traditional first-hop resilience mechanisms suffer from scalability limitations.

The alternative solution is Multi-homed EVPN Head End that, in analogy, is the EVPN flavor of Pseudo-Wire Head End (PWHE). Multi-homed EVPN Head End allows the termination of Access pseudowires or PWs (like EVPN-VPWS) into a Layer 3 [virtual routing and forwarding (VRF) or global] domain. PWHE subinterface resides in customer VRFs allowing service providers to offer IP services such as DHCP, NTP, and Layer 3 VPN for internet connectivity.

Multi-homed EVPN Head End has the following advantages:

- Decouples 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 UNI interface set.
- Extends and expands service provider's Layer 3 service footprints.
- Allows provisioning features such as QoS and ACL, L3VPN on a per PWHE subinterface

The Multi-homed EVPN headend solution supports redundant Layer 3 gateway functionality over PW-Ether interface termination, residing on a pair of redundant PE routers. The PW-Ether subinterfaces offer redundancy in the Core and first-hop router towards the access on a per-customer-service basis.

Multi-homed EVPN is supported with:

- Regular attachment circuits
- Physical Ethernet ports
- Bundle interfaces
- PW-Ether interfaces

Multi-homed EVPN headend supports three load balancing modes. These load balancing modes apply to the access side only.

- Single-Active (Layer 2 Access), also referred to as anycast single-active mode. That is, all-active in Layer 3 Core and single-active in Layer 2 Access, which is the default load-balancing mode for PWHE.
- All-Active
- Port-Active

Note The load balancing mode for the EVPN headend at the core is always ALL-ACTIVE and cannot be modified.

As part of the Multi-Homed EVPN Headend functionality, a new syntax is added under the interface (EVPN) command as shown in the following example:

```
router(config)#evpn interface ?
  PW-Ether              PWHE Ethernet Interface | short name is PE
```

For details about the interface (EVPN) command, see the *VPN and Ethernet Services Command Reference for Cisco ASR 9000 Series Routers*.

How to Implement Point to Point Layer 2 Services

This section describes the tasks required to implement Point to Point Layer 2 Services:

Configuring an Interface or Connection for Point to Point Layer 2 Services

Perform this task to configure an interface or a connection for Point to Point Layer 2 Services.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **l2transport**
4. **exit**
5. **interface** *type interface-path-id*
6. Use the **commit** or **end** command.
7. **show interface** *type interface-id*

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and configures an interface.

Step 3 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# l2transport
```

Enables L2 transport on the selected interface.

Step 4 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-if-l2)# exit
```

Exits the current configuration mode.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/0/0/0
```

Enters interface configuration mode and configures an interface.

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

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

end - Prompts user to take one of these actions:

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

Step 7 **show interface** *type interface-id*

Example:

```
RP/0/RSP0/CPU0:router show interface TenGigE 0/0/0/0
```

(Optional) Displays the configuration settings you committed for the interface.

Configuring Local Switching

Perform this task to configure local switching.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **interface** *type interface-path-id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

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

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface TenGigE 0/7/0/6.5
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: Gigabit Ethernet/IEEE 802.3 interfaces
- TenGigE: TenGigabit Ethernet/IEEE 802.3 interfaces
- CEM: Circuit Emulation interface

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/4/0/30
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: Gigabit Ethernet/IEEE 802.3 interfaces
- TenGigE: TenGigabit Ethernet/IEEE 802.3 interfaces

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

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

end - Prompts user to take one of these actions:

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

Configuring Local Connection Redundancy

Perform this task to configure local connection redundancy.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **backup interface** *type interface-path-id*
6. **interface** *type interface-path-id*
7. **interface** *type interface-path-id*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

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

Step 5 **backup interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# backup interface Bundle-Ether 0/7/0/6.5
```

Configures local connect redundancy.

Note The attachment circuit (AC) must be bundle interface that is part of MC-LAG. The backup interfaces can either be bundle or Ethernet port.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Bundle-Ether 0/7/0/6.2
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: Gigabit Ethernet/IEEE 802.3 interfaces
- TenGigE: TenGigabit Ethernet/IEEE 802.3 interfaces
- CEM: Circuit Emulation interface

Step 7 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Bundle-Ether 0/7/0/6.1
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: Gigabit Ethernet/IEEE 802.3 interfaces
- TenGigE: TenGigabit Ethernet/IEEE 802.3 interfaces

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

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

end - Prompts user to take one of these actions:

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

Configuring Static Point-to-Point Cross-Connects

Note Consider this information about cross-connects when you configure static point-to-point cross-connects:

- An cross-connect is uniquely identified with the pair; the cross-connect name must be unique within a group.
- A segment (an attachment circuit or pseudowire) is unique and can belong only to a single cross-connect.
- A static VC local label is globally unique and can be used in one pseudowire only.
- No more than 16,000 cross-connects can be configured per router.

Note Static pseudowire connections do not use LDP for signaling.

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*
7. **mpls static label local** { *value* } **remote** { *value* }
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

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

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/9
```

Specifies the interface type and instance.

Step 6 **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.2.2.2 pw-id 2000
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix.

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN.

Step 7 **mpls static label local** { *value* } **remote** { *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# mpls static label local 699 remote 890
```

Configures local and remote label ID values.

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

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

end - Prompts user to take one of these actions:

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

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

Configuring 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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

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

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: GigabitEthernet/IEEE 802.3 interfaces.
- TenGigE: TenGigabitEthernet/IEEE 802.3 interfaces.
- CEM: Circuit Emulation interface

Step 6 **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000
```

Configures the pseudowire segment for the cross-connect.

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN.

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

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

end - Prompts user to take one of these actions:

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

Configuring Inter-AS

The Inter-AS configuration procedure is identical to the L2VPN cross-connect configuration tasks (see "Configuring Static Point-to-Point Cross-Connects" section and "Configuring Dynamic Point-to-Point Cross-Connects" section) except that the remote PE IP address used by the cross-connect configuration is now reachable through iBGP peering.

Note You must be knowledgeable about IBGP, EBGP, and ASBR terminology and configurations to complete this configuration.

Configuring L2VPN Quality of Service

This section describes how to configure L2VPN quality of service (QoS) in port mode and mode, VLAN mode, Frame Relay and ATM sub-interfaces.

Restrictions

The **l2transport** command cannot be used with any IP address, L3, or CDP configuration.

Configuring an L2VPN Quality of Service Policy in Port Mode

This procedure describes how to configure an L2VPN QoS policy in port mode.

Note In port mode, the interface name format does not include a subinterface number; for example, GigabitEthernet0/1/0/1.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **l2transport**
4. **service-policy** [**input** | **output**] [*policy-map-name*]
5. Use the **commit** or **end** command.
6. **show qos interface** *type interface-id* **service-policy** [**input** | **output**] [*policy-map-name*]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the configuration mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet 0/0/0/0
```

Specifies the interface attachment circuit.

Step 3 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# l2transport
```

Configures an interface or connection for L2 switching.

Step 4 **service-policy** [**input** | **output**] [*policy-map-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-if)# service-policy input servpol1
```

Attaches a QoS policy to an input or output interface to be used as the service policy for that interface.

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

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

end - Prompts user to take one of these actions:

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

Step 6 **show qos interface** *type interface-id* **service-policy** [**input** | **output**] [*policy-map-name*]

Example:

```
RP/0/RSP0/CPU0:router# show qos interface gigabitethernet 0/0/0/0 input serpol1
```

(Optional) Displays the QoS service policy you defined.

Configuring an L2VPN Quality of Service Policy in VLAN Mode

This procedure describes how to configure a L2VPN QoS policy in VLAN mode.

Note In VLAN mode, the interface name must include a subinterface. For example: GigabitEthernet0/1/0/1.1. The l2transport command must follow the interface type on the same CLI line. For example: interface GigabitEthernet 0/0/0/0.1 l2transport".

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id.subinterface* **l2transport**
3. **service-policy** [**input** | **output**] [*policy-map-name*]
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface** *type interface-path-id.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/0/0/0.1 l2transport
```

Configures an interface or connection for L2 switching.

Note In VLAN Mode, you must enter the **l2transport** keyword on the same line as the interface.

Step 3 **service-policy** [**input** | **output**] [*policy-map-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-if)# service-policy input servpol1
```

Attaches a QoS policy to an input or output interface to be used as the service policy for that interface.

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

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

end - Prompts user to take one of these actions:

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

Configuring Multisegment Pseudowire

This section describes these tasks:

Provisioning a Multisegment Pseudowire Configuration

Configure a multisegment pseudowire as a point-to-point (p2p) cross-connect. For more information on P2P cross-connects, see the “Configuring Static Point-to-Point Cross-Connects".

SUMMARY STEPS

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

8. **neighbor** *A.B.C.D* **pw-id** *value*
9. **pw-class** *class-name*
10. **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

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

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group MS-PW1
```

Configures a cross-connect group name using a free-format 32-character string.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p ms-pw1
```

Enters P2P configuration submode.

Step 5 **neighbor** *A.B.C.D* **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.165.200.25 pw-id 100
```

Configures a pseudowire for a cross-connect.

The IP address is that of the corresponding PE node.

The **pw-id** must match the **pw-id** of the PE node.

Note For MSPW cross-connect configuration is performed at local PE, S-PE and Remote-PE.

Step 6 **pw-class** *class-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# exit
```

Exits pseudowire class submode and returns the router to the parent configuration mode.

Step 8 **neighbor** *A.B.C.D* **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.165.202.158 pw-id 300
```

Configures a pseudowire for a cross-connect.

The IP address is that of the corresponding PE node.

The **pw-id** must match the **pw-id** of the PE node.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 10 **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit
```

Saves configuration changes to the running configuration file and remains in the configuration session.

Provisioning a Global Multisegment Pseudowire Description

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

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**

3. **description** *value*
4. **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **description** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# description S-PE1
```

Populates the Pseudowire Switching Point TLV. This TLV records all the switching points the pseudowire traverses.

Each multisegment pseudowire can have its own description. If it does not have one, this global description is used.

Step 4 **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# commit
```

Saves configuration changes to the running configuration file and remains in the configuration session.

Provisioning a Cross-Connect Description

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

SUMMARY STEPS

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

6. **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group MS-PW1
```

Configures a cross-connect group name using a free-format 32-character string.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p ms-pw1
```

Enters P2P configuration submode.

Step 5 **description** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# description MS-PW from T-PE1 to T-PE2
```

Populates the Pseudowire Switching Point TLV. This TLV records all the switching points the pseudowire traverses.

Each multisegment pseudowire can have its own description. If it does not have one, a global description is used. For more information, see the "Provisioning a Multisegment Pseudowire Configuration".

Step 6 **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# commit
```

Saves configuration changes to the running configuration file and remains in the configuration session.

Provisioning Switching Point TLV Security

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

Virtual Circuit Connection Verification (VCCV) may not work on multisegment pseudowires with the **switching-tlv** parameter set to “hide”. For more information on VCCV, see the “Virtual Circuit Connection Verification on L2VPN".

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class dynamic_mpls
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 4 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation mpls
```

Sets pseudowire encapsulation to MPLS.

Step 5 **protocol ldp**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# protocol ldp
```

Sets pseudowire signaling protocol to LDP.

Step 6 **switching-tlv hide**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# switching-tlv hide
```

Sets pseudowire TLV to hide.

Step 7 **commit**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)#commit
```

Saves configuration changes to the running configuration file and remains in the configuration session.

Enabling Multisegment Pseudowires

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 3 **pw-status**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-status
```

Enables all pseudowires configured on this Layer 2 VPN.

Note Use the **pw-status disable** command to disable pseudowire status.

Step 4 **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# commit
```

Saves configuration changes to the running configuration file and remains in the configuration session.

Configuring Pseudowire Redundancy

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

These topics describe how to configure pseudowire redundancy:

Configuring Point-to-Point Pseudowire Redundancy

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

SUMMARY STEPS

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

11. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
 RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/O/RSP0/CPU0:router(config-l2vpn)# pw-class path1
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)#
```

Configures the pseudowire class name.

Step 4 **backup disable {delay value | never}**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20
```

This command specifies how long the primary pseudowire should wait after it becomes active to take over from the backup pseudowire.

- Use the **delay** keyword to specify the number of seconds that elapse after the primary pseudowire comes up before the secondary pseudowire is deactivated. The range is from 0 to 180.
- Use the **never** keyword to specify that the secondary pseudowire does not fall back to the primary pseudowire if the primary pseudowire becomes available again, unless the secondary pseudowire fails.

Step 5 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# exit
RP/O/RSP0/CPU0:router(config-l2vpn)#
```

Exits the current configuration mode.

Step 6 **xconnect group** `group-name`

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group A
RP/0/RSP0/CPU0:router(config-l2vpn-xc)#
```

Enters the name of the cross-connect group

Step 7 **p2p {xconnect-name}**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p xc1
```

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

Step 8 **neighbor A.B.C.D pw-id value**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 2
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire segment for the cross-connect.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)#pw-class path1
```

Configures the pseudowire class name.

Step 10 **backup {neighbor A.B.C.D} {pw-id value}**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#
```

Configures the backup pseudowire for the cross-connect.

- Use the **neighbor** keyword to specify the peer to the cross-connect. The A.B.C.D argument is the IPv4 address of the peer.
- Use the **pw-id** keyword to configure the pseudowire ID. The range is from 1 to 4294967295.

Step 11 **end** or **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#end
```

or

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  - Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Forcing a Manual Switchover to the Backup Pseudowire

To force the router to switch over to the backup or 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.

Configuring a Backup Pseudowire

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **neighbor** *ip-address* **pw-id** *value*
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group A
RP/0/RSP0/CPU0:router(config-l2vpn-xc)#
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 2
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire segment for the cross-connect.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 11
```

Configures the backup pseudowire for the cross-connect.

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

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

end - Prompts user to take one of these actions:

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

Configuring Point-to-Point Pseudowire Redundancy

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **pw-class** { *class-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class path1
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)#
```

Configures the pseudowire class name.

Step 4 **backup disable** { **delay***value* | **never** }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20
```

This command specifies how long the primary pseudowire should wait after it becomes active to take over for the backup pseudowire.

- Use the **delay** keyword to specify the number of seconds that elapse after the primary pseudowire comes up before the secondary pseudowire is deactivated. The range, in seconds, is from 0 to 180.

- Use the **never** keyword to specify that the secondary pseudowire does not fall back to the primary pseudowire if the primary pseudowire becomes available again, unless the secondary pseudowire fails.

Step 5 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# exit
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Exits the current configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group A
RP/0/RSP0/CPU0:router(config-l2vpn-xc)#
```

Enters the name of the cross-connect group.

Step 7 **p2p** { *xconnect-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p xc1
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#
```

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

Step 8 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 2
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire segment for the cross-connect.

Step 9 **pw-class** { *class-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class path1
```

Configures the pseudowire class name.

Step 10 **backup** { **neighbor** *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#
```

Configures the backup pseudowire for the cross-connect.

- Use the **neighbor** keyword to specify the peer to the cross-connect. The A.B.C.D argument is the IPv4 address of the peer.
- Use the **pw-id** keyword to configure the pseudowire ID. The range is from 1 to 4294967295.

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

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

end - Prompts user to take one of these actions:

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

Forcing a Manual Switchover to the Backup Pseudowire

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

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

Configuring Preferred Tunnel Path

This procedure describes how to configure a preferred tunnel path.

Note The tunnel used for the preferred path configuration is an MPLS Traffic Engineering (MPLS-TE) tunnel.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** {*name*}
4. **encapsulation mpls**
5. **preferred-path** {**interface**} {**tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value*} [**fallback disable**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the configuration mode.

Step 2 **l2vpn**

Example:

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

Enters L2VPN configuration mode.

Step 3 **pw-class** {*name*}

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# pw-class path1
```

Configures the pseudowire class name.

Step 4 **encapsulation mpls**

Example:

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

Configures the pseudowire encapsulation to MPLS.

Step 5 **preferred-path** {**interface**} {**tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value*} [**fallback disable**]

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te 11 fallback
disable
```

Configures preferred path tunnel settings. If the fallback disable configuration is used and once the TE/TP tunnel is configured as the preferred path goes down, the corresponding pseudowire can also go down.

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

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

end - Prompts user to take one of these actions:

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

Configuring PW Status OAM

Perform this task to configure pseudowire status OAM.

SUMMARY STEPS

1. configure
2. l2vpn
3. pw-oam refresh transmit seconds
4. endor**commit**

DETAILED STEPS

Step 1 configure

Example:

```
RP/0/RSP0RP0/CPU0:router# configure
```

Enters the configuration mode.

Step 2 l2vpn

Example:

```
RP/0/RSP0RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 pw-oam refresh transmit seconds

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# pw-oam refresh transmit 100
```

Enables pseudowire OAM functionality..

Note The refresh transmit interval ranges from 1 to 40 seconds.

Step 4 endor**commit**

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# end
```

or

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# commit
```

Saves configuration changes.

- When you issue the end command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling Flow-based Load Balancing

Perform this task to enable flow-based load balancing.

SUMMARY STEPS

1. configure
2. l2vpn
3. load-balancing flow {src-dst-mac | src-dst-ip}
4. endor**commit**

DETAILED STEPS

Step 1 configure

Example:

```
RP/0/RSP0RP0/CPU0:router# configure
```

Enters the configuration mode.

Step 2 l2vpn

Example:

```
RP/0/RSP0RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 load-balancing flow {src-dst-mac | src-dst-ip}

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# load-balancing flow src-dst-ip
```

Enables flow based load balancing for all the pseudowires and bundle EFPs under L2VPN, unless otherwise explicitly specified for pseudowires via pseudowire class and bundles via EFP-hash.

Step 4 endor**commit**

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# end
```

or

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling Flow-based Load Balancing for a Pseudowire Class

Perform this task to enable flow-based load balancing for a pseudowire class.

SUMMARY STEPS

1. configure
2. l2vpn
3. pw-class {name}
4. encapsulation mpls
5. load-balancing pw-label
6. endor**commit**

DETAILED STEPS

Step 1 configure

Example:

```
RP/0/RSP0RP0/CPU0:router# configure
```

Enters the configuration mode.

Step 2 l2vpn

Example:

```
RP/0/RSP0RP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 pw-class {name}

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn)# pw-class path1
```

Configures the pseudowire class name.

Step 4 encapsulation mpls

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
```

Configures the pseudowire encapsulation to MPLS.

Step 5 load-balancing pw-label

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# load-balancing pw-label
```

Enables all pseudowires using the defined class to use virtual circuit based load balancing.

Step 6 endor**commit**

Example:

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# end
```

or

```
RP/0/RSP0RP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Enabling Pseudowire Grouping

Perform this task to enable pseudowire grouping.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

Step 3 **pw-grouping**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-grouping
```

Enables pseudowire grouping

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

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

end - Prompts user to take one of these actions:

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

Setting Up Your Multicast Connections

Refer to the *Implementing Multicast Routing on Cisco ASR 9000 Series Aggregation Services Routers* module of the *Cisco ASR 9000 Series Aggregation Services Router Multicast Configuration Guide* and the *Multicast Routing and Forwarding Commands on Cisco ASR 9000 Series Aggregation Services Routers* module of the *Cisco ASR 9000 Series Aggregation Services Router Multicast Command Reference*.

SUMMARY STEPS

1. configure
2. multicast-routing [address-family ipv4]
3. interface all enable
4. exit
5. router igmp
6. version {1 | 2 | 3}
7. endor**commit**
8. show pim [ipv4] group-map **[ip-address-name] [info-source]**
9. show pim [vrf **vrf-name]** [ipv4] topology **[source-ip-address [group-ip-address]** | entry-flag **flag** | interface-flag | summary] [route-count]

DETAILED STEPS

Step 1 configure

Example:

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

Enters global configuration mode.

Step 2 multicast-routing [address-family ipv4]

Example:

```
RP/0/RSP0/CPU0:router(config)# multicast-routing
```

Enters multicast routing configuration mode.

- These multicast processes are started: **MRIB, MFWD, PIM,** and **IGMP**.
- For **IPv4**, **IGMP** version 3 is enabled by default.

- For IPv4, use the

  ```
  address-family ipv4
  ```

  keywords

Step 3 interface all enable

Example:

```
RP/0/RSP0/CPU0:router(config-mcast-ipv4)# interface all enable
```

Enables multicast routing and forwarding on all new and existing interfaces.

Step 4 exit

Example:

```
RP/0/RSP0/CPU0:router(config-mcast-ipv4)# exit
```

Exits multicast routing configuration mode, and returns the router to the parent configuration mode.

Note For Leaf PEs, if you intend to enable IGMPSN on the bridge domain, ensure that you configure internal querier inside the IGMPSN profile.

Step 5 router igmp

Example:

```
RP/0/RSP0/CPU0:router(config)# router igmp
```

(Optional) Enters router IGMP configuration mode.

Step 6 version {1 | 2 | 3}

Example:

```
RP/0/RSP0/CPU0:router(config-igmp)# version 3
```

(Optional) Selects the IGMP version that the router interface uses.

- The default for IGMP is version 3.
- Host receivers must support IGMPv3 for PIM-SSM operation.
- If this command is configured in router IGMP configuration mode, parameters are inherited by all new and existing interfaces. You can override these parameters on individual interfaces from interface configuration mode.

Step 7 endor**commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# end
```

or

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Step 8 show pim [ipv4] group-map **`[ip-address-name] [info-source]`**

Example:

```
RP/0//CPU0:router# show pim ipv4 group-map
```

(Optional) Displays group-to-**PIM** mode mapping.

Step 9 show pim [vrf **`vrf-name]`** [ipv4] topology **`[source-ip-address [group-ip-address]`** | entry-flag **`flag`** | interface-flag | summary] [route-count]

Example:

```
RP/0/RSP0/CPU0:router# show pim topology
```

(Optional) Displays PIM topology table information for a specific group or all groups.

Configuring AToM IP Interworking

Perform this task to configure AToM IP Interworking.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group***group-name*
4. **p2p***xconnect-name*
5. **interworking ipv4**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

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

Step 5 **interworking ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

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

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

end - Prompts user to take one of these actions:

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

Configuring PPP IP Interworking

Perform this tasks to configure PPP IP Interworking:

SUMMARY STEPS

1. configure
2. interface type **`interface-path-id`**
3. encapsulation ppp
4. l2transport
5. end
6. l2vpn
7. xconnect group **`group-name`**
8. p2p **`xconnect-name`**
9. interface type **`interface-path-id`**
10. interface type **`interface-path-id`**
11. interworking ipv4
12. interface type **`interface-path-id`**
13. neighbor**`A.B.C.D`pw-id**
14. pw-class **`interface-path-id`**
15. exit
16. interworking ipv4

17. endor**commit**

DETAILED STEPS

Step 1 configure

Example:

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

Enters global configuration mode.

Step 2 interface type **interface-path-id**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Serial0/2/1/0/1/1/1:0
```

Specifies the interface type and instance.

Step 3 encapsulation ppp

Example:

```
RP/0/RSP0/CPU0:router(config-if)# encapsulation ppp
```

Sets encapsulation type to PPP.

Step 4 l2transport

Example:

```
RP/0/RSP0/CPU0:router(config-if)# l2transport
```

Enables Layer 2 transport on the selected interface.

Step 5 end

Example:

```
RP/0/RSP0/CPU0:router(config-if-l2)# end
```

Returns to global configuration mode.

Step 6 l2vpn

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 7 xconnect group **group-name**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

Step 8 p2p **xconnect-name**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p 1
```

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

Step 9 interface type **`interface-path-id`**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Serial0/2/1/0/1/1/1:0
```

Specifies the interface type and instance.

Step 10 interface type **`interface-path-id`**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1
```

Specifies the interface type and instance.

Step 11 interworking ipv4

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

Step 12 interface type **`interface-path-id`**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1
```

Specifies the interface type and instance.

Step 13 neighbor**`A.B.C.D`pw-id**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Serial0/0/0/0/2/1/1:0
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN

Step 14 pw-class **`interface-path-id`**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# pw-class class_c1
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 15 exit

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# exit
```

Exits the current configuration mode.

Step 16 interworking ipv4

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

Step 17 endor**commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# end
```

or

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IP Interworking between PPP and Ethernet

Perform this tasks to configure PPP IP Interworking:

SUMMARY STEPS

1. **configure**
2. **interface type***interface-path-id*
3. **l2transport**
4. **end**
5. **l2vpn**
6. **xconnect group***group-name*
7. **p2p***xconnect-name*
8. **interface type***interface-path-id*
9. **interface type** *interface-path-id*
10. **interworking ipv4**
11. **interface type** *interface-path-id*
12. **neighbor***A.B.C.D***pw-id**
13. **pw-class***class-name*
14. **pw-class***class-name*
15. **exit**
16. **interworking ipv4**
17. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Serial0/2/1/0/1/1/1:0
```

Specifies the interface type and instance.

Step 3 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# l2transport
```

Enables Layer 2 transport on the selected interface.

Step 4 **end**

Example:

```
RP/0/RSP0/CPU0:router(config-if-l2)# end
```

Returns to global configuration mode.

Step 5 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

Step 7 **p2p***xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p 1
```

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

Step 8 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Serial0/2/1/0/1/1/1:0
```

Specifies the interface type and instance.

Step 9 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1
```

Specifies the interface type and instance.

Step 10 **interworking ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

Step 11 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Serial0/0/0/0/2/1/1:0
```

Specifies the interface type and instance.

Step 12 **neighbor***A.B.C.D***pw-id**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Serial0/0/0/0/2/1/1:0
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN

Step 13 **pw-class***class-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Serial0/0/0/0/2/1/1:0
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN

Step 14 **pw-class***class-name*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# pw-class class_cem
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 15 **exit**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# exit
```

Exits the current configuration mode.

Step 16 **interworking ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

Step 17 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring MLPPP IP Interworking

Perform this tasks to configure cHDLC IP Interworking:

SUMMARY STEPS

1. **configure**
2. **interface type***interface-path-id*
3. **multilink [fragment|interleave|ncp]**
4. **l2transport**
5. **end**
6. **l2vpn**
7. **xconnect group** *group-name*
8. **p2p** *xconnect-name*
9. **interface type***interface-path-id*
10. **interface type***interface-path-id*
11. **interworking ipv4**
12. **interface type***interface-path-id*
13. **neighbor***{A.B.C.D}***{pw-id***value}*
14. **pw-class***class-name*
15. **exit**
16. **interworking ipv4**
17. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Multilink0/2/1/0/1
```

Specifies the interface type and instance.

Step 3 **multilink [fragment|interleave|ncp]**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# multilink
```

Modifies multilink parameters

Step 4 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-if-multilink)# l2transport
```

Enables Layer 2 transport on the selected interface.

Step 5 **end**

Example:

```
RP/0/RSP0/CPU0:router(config-if-l2)# end
```

Returns to global configuration mode.

Step 6 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 7 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

Step 8 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p 1
```

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

Step 9 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface Serial0/2/1/0/1/1/1:0
```

Specifies the interface type and instance.

Step 10 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1
```

Specifies the interface type and instance.

Step 11 **interworking ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

Step 12 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface Serial0/0/0/0/2/1/1:0
```

Specifies the interface type and instance.

Step 13 **neighbor***{A.B.C.D}***{pw-id***value}*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 120.120.120.120 pw-id 3
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN

Step 14 **pw-class***class-name*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# pw-class class_cem
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 15 **exit**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# exit
```

Exits the current configuration mode.

Step 16 **interworking ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interworking ipv4
```

Configures IPv4 interworking under P2P.

Step 17 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring Circuit Emulation Over Packet Switched Network

Perform these tasks to configure CEoP:

Adding CEM attachment circuit to a Pseudowire

Perform this task to add a CEM attachment circuit to a pseudowire.:

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface type** *interface-path-id*
6. **neighbor** *A.B.C.D* **pw-id**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

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

Step 5 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface CEM0/1/0/9:10
```

Specifies the interface type and instance.

Step 6 **neighbor** *A.B.C.D* **pw-id**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 120.120.120.120 pw-id 3
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN

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

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

end - Prompts user to take one of these actions:

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

Associating a Pseudowire Class

Perform this task to associate the attachment circuit with a pseudowire class.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation mpls**
5. **protocol ldp**
6. **end**
7. **xconnect group** *group-name*
8. **p2p** *xconnect-name*
9. **interface** *type interface-path-id*
10. **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*
11. **pw-class** *class-name*
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# pw-class class_cem
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 4 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation mpls
```

Sets pseudowire encapsulation to MPLS.

Step 5 **protocol ldp**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# protocol ldp
```

Sets pseudowire signaling protocol to LDP.

Step 6 **end**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-encap-mpls)# end
```

System prompts you to commit changes:

```
Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
```

- • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
  • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
  • Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.

Step 7 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Configures a cross-connect group.

Step 8 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

Configures a point-to-point cross-connect.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface CEM0/1/0/9:20
```

Specifies the interface type and instance.

Step 10 **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.2.2.2 pw-id 11
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix.

Optionally, you can disable the control word or set the transport-type to Ethernet or VLAN.

Note Pseudowire status (pw-status) is enabled by default, use the **pw-status disable** command to disable pseudowire status if required.

Step 11 **pw-class** *class-name*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# pw-class class_cem
```

Associates the P2P attachment circuit with the specified pseudowire class.

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

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

end - Prompts user to take one of these actions:

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

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

Enabling Pseudowire Status

Perform this task to enable pseudowire status.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 configure

Example:

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

Enters global configuration mode.

Step 2 l2vpn

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 3 pw-status

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# pw-status
```

Enables all pseudowires configured on this Layer 2 VPN.

Note Use the pw-status disable command to disable pseudowire status.

Step 4 commit

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# commit
```

Saves configuration changes to the running configuration file and remains in the configuration session.

Configuring a Backup Pseudowire

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **neighbor** *ip-address* **pw-id** *value*
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group A
RP/0/RSP0/CPU0:router(config-l2vpn-xc)#
```

Enters the name of the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.0.0.1 pw-id 2
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)#
```

Configures the pseudowire segment for the cross-connect.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 11
```

Configures the backup pseudowire for the cross-connect.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2VPN Nonstop Routing

Perform this task to configure L2VPN Nonstop Routing.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **nsr**
4. **logging nsr**
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the Global Configuration mode.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# nsr
```

Enables L2VPN nonstop routing.

Step 4 **logging nsr**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# logging nsr
```

Enables logging of NSR events.

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

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

end - Prompts user to take one of these actions:

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

Configure MPLS LDP Nonstop Routing

Perform this task to enable Label Distribution Protocol (LDP) Nonstop Routing (NSR) for synchronizing label information between active and standby LDPs. From Release 6.1.1 onwards, with the introduction of stateful LDP feature, you must explicitly configure LDP NSR to synchronize label information between active and standby LDPs.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **nsr**
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **mpls ldp**

Example:

```
RP/0/RSP0/CPU0:router(config)# mpls ldp
```

Enters MPLS LDP configuration mode.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# nsr
```

Enables LDP nonstop routing.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2TPv3 over IPv6 Tunnels

Perform these tasks to configure the L2TPv3 over IPv6 tunnels:

Configuring Neighbor AFI for Pseudowire

Perform this task to configure the neighbor AFI for pseudowire.

Restriction L2TPv3 over IPv6Tunnels is supported only on layer 2 transport sub-interfaces and not physical interfaces.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **neighbor ipv6** X:X::X **pw-id***pseudowire-id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp_1
```

Configures a cross-connect group and specifies it's name.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
```

Configures a point-to-point cross-connect.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/4/0/30
```

Specifies the interface type ID. The choices are:

- GigabitEthernet: Gigabit Ethernet/IEEE 802.3 interfaces
- TenGigE: TenGigabit Ethernet/IEEE 802.3 interfaces

Step 6 **neighbor ipv6** X:X::X **pw-id***pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor ipv6 1111:2222::cdef pw-id 2000
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2TPv3 encapsulation and protocol

Perform this task to configure L2TPv3 encapsulation and protocol.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation l2tpv3**
5. **protocol l2tpv3**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# pw-class l2tpv3_class
```

Enters pseudowire class submode, allows a pseudowire class template definition.

These keywords can be configured in the pseudowire class (pw-class) configuration mode; however, the keywords are not applicable for the over L2TPv3 over IPv6 tunnels:

- **cookie**

- **dfbit**
- **ipv4 source**
- **pmtu**
- **sequencing**
- **transport-mode**

Step 4 **encapsulation l2tpv3**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation l2tpv3
```

Sets pseudowire encapsulation to L2TPv3

Step 5 **protocol l2tpv3**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# protocol l2tpv3
```

Sets pseudowire signaling protocol to L2TPv3.

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

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

end - Prompts user to take one of these actions:

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

Configuring Source IPv6 address for L2TPv3 over IPv6 tunnel

Perform this task to configure the source IPv6 address for the L2TPv3 over IPv6 tunnel.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface type** *interface-path-id*
6. **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*
7. **source** *pw-source-address*
8. **end** or **commit**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# xconnect group g1
```

Configures the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc)# p2p xc3
```

Configures the point-to-point cross-connect.

Step 5 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/4.2
```

Specifies the interface type ID.

Step 6 **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# neighbor ipv6 1111:2222::cdef pw-id 1
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

Step 7 **source** *pw-source-address*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# source 1111:2222::abcd
```

Configures the source IPv6 address of the pseudowire.

Note The source IPv6 address must be unique and chosen arbitrarily. It should not be configured on any type of interfaces in the router.

Step 8 **end** or **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# end
```

or

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Local and Remote Sessions

Perform this task to configure local and remote sessions.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface type** *interface-path-id*
6. **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*
7. **l2tp static local session** *session-id*
8. **l2tp static remote session** *session-id*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# xconnect group g1
```

Configures the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc)# p2p xc3
```

Configures the point-to-point cross-connect.

Step 5 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/4.2
```

Specifies the interface type ID.

Step 6 **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# neighbor ipv6 1111:2222::cdef pw-id 1
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

Step 7 **l2tp static local session** *session-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# l2tp static local session 1
```

(Optional) Configures a static local session for the L2TP pseudowire.

Note If you configure the local session ID, it is ignored for decapsulation-side processing by the ASR9000 Series routers.

Step 8 **l2tp static remote session** *session-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# l2tp static remote session 1
```

(Optional) Configures a static remote session for the L2TP pseudowire.

Note When configured, remote session value(expected at the decapsulation side) is used for encapsulation- side processing, and the value in the session value field of the L2TPv3 header is programmed.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring Local And Remote Cookies

Perform this task to configure local and remote cookies.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface type** *interface-path-id*
6. **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*
7. **l2tp static local cookie size** *bytes*
8. **l2tp static local cookie size** *bytes*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# xconnect group g1
```

Configures the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc)# p2p xc3
```

Configures the point-to-point cross-connect.

Step 5 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/4.2
```

Specifies the interface type ID.

Step 6 **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor ipv6 1111:2222::cdef pw-id 1
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

Step 7 **l2tp static local cookie size** *bytes*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# l2tp static local cookie size 0
```

Configures the static local cookie size settings for the L2TP pseudowire.

Note In the case of a non-zero cookie size, the value of the cookie is a mandatory argument.

Step 8 **l2tp static local cookie size** *bytes*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# l2tp static remote cookie size 0
```

Configures the static remote cookie size settings for the L2TP pseudowire.

Note In the case of a non-zero cookie size, the value of the cookie is a mandatory argument.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

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

Enabling L2TP Static Submode

Perform this task to enable L2TP static submode.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface type** *interface-path-id*
6. **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*
7. **l2tp static**
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# xconnect group g1
```

Configures the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc)# p2p xc3
```

Configures the point-to-point cross-connect.

Step 5 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/4.2
```

Specifies the interface type ID.

Step 6 **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# neighbor ipv6 1111:2222::cdef pw-id
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

Step 7 **l2tp static**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)#
```

Enters L2TP static configuration submode.

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

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

end - Prompts user to take one of these actions:

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

Enabling TOS Reflection in the L2TPv3 Header

Perform this task to enable the type of service (TOS) reflection in the L2TPv3 header.

SUMMARY STEPS

1. configure
2. l2vpn
3. pw-class**`class-name`**
4. encapsulation l2tpv3
5. protocol l2tpv3
6. neighbor ipv6**`peer-address pw-id pseudowire-id`**
7. tos { reflect | value }
8. endor**commit**

DETAILED STEPS

Step 1 configure

Example:

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

Enters global configuration mode.

Step 2 l2vpn

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# pw-class l2tpv3_class
```

Enters pseudowire class submode, and allows a pseudowire class template definition.

Step 4 encapsulation l2tpv3

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation l2tpv3
```

Sets pseudowire encapsulation to L2TPv3.

Step 5 protocol l2tpv3

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# protocol l2tpv3
```

Sets pseudowire signaling protocol to L2TPv3.

Step 6 neighbor ipv6**`peer-address pw-id pseudowire-id`**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# neighbor ipv6 1111:2222::cdef pw-id
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

Step 7 tos { reflect | value }

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# tos reflect
```

or

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# tos value 50
```

Enables type of service (TOS) reflection. As a result, copies TOS from inner IP header to the L2TPv3 header.

Additionally, use this command to set the value of TOS for the L2TPv3 pseudowire class. The range is from 0 to 255.

Step 8 endor**commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# end
```

or

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# commit
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.

  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.
- Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring TTL for L2TPv3 over IPv6 Tunnels

Perform this task to configure time to live (TTL) for L2TPv3 over IPv6 tunnels.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation l2tpv3**
5. **protocol l2tpv3**
6. **ttl** *value*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# pw-class l2tpv3_class
```

Enters pseudowire class submode, and allows a pseudowire class template definition.

Step 4 **encapsulation l2tpv3**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation l2tpv3
```

Sets pseudowire encapsulation to L2TPv3.

Step 5 **protocol l2tpv3**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# protocol l2tpv3
```

Sets pseudowire signaling protocol to L2TPv3.

Step 6 **ttl** *value*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# ttl 50
```

Sets time to live (TTL), in node hops, to a specified value. The range is from 1 to 255.

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

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

end - Prompts user to take one of these actions:

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

Configuring Traffic Mirroring over L2TPv3 over IPv6 Tunnels

Perform this task to configure traffic mirroring over L2TPv3 over IPv6 tunnels.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group***group-name*
4. **p2p** *xconnect-name*
5. **monitor-session** *session-name*
6. **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*
7. **pw-class** *class-name*
8. **source***pw-source-address*
9. **l2tp static local cookie size***size***value***bytes*
10. **l2tp static remote cookie size***size***value***bytes*
11. Use the **commit** or **end** command.

- L2TPv3 over IPv6 concepts, see L2TPv3 over IPv6.
- Configuration examples, see Configuring L2TPv3 over IPv6 Tunnels: Example

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# xconnect group span
```

Configures the cross-connect group.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc)# p2p span-foo
```

Configures the point-to-point cross-connect.

Step 5 **monitor-session** *session-name*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# monitor-session customer-foo
```

Specifies the monitor session.

Step 6 **neighbor ipv6** *peer-address* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p)# neighbor ipv6 1111:3333::cdef pw-id 1001
```

Specifies the peer with which to cross connect, and configures the pseudowire segment for the cross-connect.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# pw-class ts
```

Enters pseudowire class submode, and allows a pseudowire class template definition.

Step 8 **source***pw-source-address*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# source 1111:3333::abcd
```

Configures the source IPv6 address of the pseudowire.

Step 9 **l2tp static local cookie size***size***value***bytes*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# l2tp static local cookie size 8 value 0xabcd 0x1234
```

Configures the static local cookie size settings for the L2TP pseudowire.

Step 10 **l2tp static remote cookie size***size***value***bytes*

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-xc-p2p-pw)# l2tp static remote cookie size 8 value 0xcdef 0x5678
```

Configures the static remote cookie size settings for the L2TP pseudowire.

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

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

end - Prompts user to take one of these actions:

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

For more information on:

- L2TPv3 over IPv6 concepts, see L2TPv3 over IPv6.
- Configuration examples, see Configuring L2TPv3 over IPv6 Tunnels: Example

Configuring L2TPv3 over IPv4 Tunnels

Restriction L2TPv3 over Ipv4 Tunnels is supported only on layer 2 transport sub-interfaces and not on physical interfaces. When an untagged traffic has to be sent through L2TPv3 over IPv4, create a sub-interface with encapsulation as untagged.

This example shows how to create a sub-interface with encapsulation as untagged:

```
interface TenGigE0/3/0/1.123 l2transport
 encapsulation untagged
```

Perform these tasks to configure the L2TPv3 over IPv4 tunnels:

Configuring a Dynamic L2TPv3 Pseudowire

Perform this task to configure a dynamic L2TPv3 pseudowire connecting to a remote IPv4 peer.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *name*
4. **p2p** *name*
5. **interface** *type interface-path-id*
6. **neighbor ipv4** *ip-address* **pw-id** *number*
7. **pw-class** *pw-class-name*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enter L2VPN configure submode.

Step 3 **xconnect group** *name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group L2TPV3_V4_XC_GRP
```

Enter a name for the cross-connect group.

Step 4 **p2p** *name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p L2TPV3_P2P_1
```

Enters p2p configuration submode to configure point-to-point cross-connects.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/2/0/0/0.1
```

Specifies the interface type ID. The choices are:

- GigabitEthernet
- TenGigE

Step 6 **neighbor ipv4** *ip-address* **pw-id** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor ipv4 26.26.26.26 pw-id 100
```

Configures a pseudowire for a cross-connect.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class L2TPV3_V4_CLASS
```

Enters pseudowire class submode to define a name for the cross-connect.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2TPv3 Encapsulation and Protocol

Perform this task to configure L2TPv3 encapsulation and protocol.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation l2tpv3**
5. **protocol l2tpv3**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router (config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn)# pw-class l2tpv3_class
```

Enters pseudowire class submode, allows a pseudowire class template definition.

These keywords can be configured in the pseudowire class (pw-class) configuration mode:

- **cookie**
- **dfbit**
- **ipv4 source**

Step 4 **encapsulation l2tpv3**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc)# encapsulation l2tpv3
```

Sets pseudowire encapsulation to L2TPv3

Step 5 **protocol l2tpv3**

Example:

```
RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-l2tpv3)# protocol l2tpv3
```

Sets pseudowire signaling protocol to L2TPv3.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2TP Control-Channel Parameters

L2TP control-channel parameters are used in control-channel authentication, keepalive messages, and control-channel negotiation. In a L2tpv3 session, the same L2TP class must be configured on both PE routers.

The following L2TP control-channel parameters can be configured in L2TP class configuration mode:

- Authentication for the L2TP control-channel
- Password used for L2TP control-channel authentication
- Retransmission parameters used for control messages.
- Timeout parameters used for the control-channel.
- Maintenance Parameters
- L2TPv3 Control Message Hashing

Perform this task to create a template of L2TP control-channel parameters that can be inherited by different pseudowire classes.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *l2tp-class-name*
3. **authentication**
4. **password** {**0** | **7**} *password*
5. **retransmit** { **initial retries** *initial-retries* | **retries** *retries* | **timeout** { **max** | **min** } *timeout* }
6. **hello-interval** *interval*
7. **digest** { **check disable** | **hash** { **MD5** | **SHA1** }] | **secret** { **0** | **7** } *password*]
8. **hidden**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2tp-class** *l2tp-class-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# l2tp-class L2TP-CLASS
```

Specifies the L2TP class name and enters L2TP class configuration mode.

Step 3 **authentication**

Example:

```
RP/0/RSP0/CPU0:router(config-l2tp-class)# authentication
```

Enables authentication for the control-channel between PE routers.

Step 4 **password** {**0** | **7**} *password*

Example:

```
RP/0/RSP0/CPU0:router(config-l2tp-class)# password 7 pwd_1
```

Configures the password used for control-channel authentication.

- **[0 | 7]**—Specifies the input format of the shared secret. The default value is 0.
  - **0**—Specifies an encrypted password will follow.
  - **7**—Specifies an unencrypted password will follow.
- *password*—Defines the shared password between peer routers.

Step 5 **retransmit** { **initial retries** *initial-retries* | **retries** *retries* | **timeout** { **max** | **min** } *timeout* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2tp-class)# retransmit retries 10
```

Configures parameters that affect the retransmission of control packets.

- **initial retries**—Specifies how many SCCRQs are re-sent before giving up on the session. Range is 1 to 1000. The default is 2.
- **retries**—Specifies how many retransmission cycles occur before determining that the peer PE router does not respond. Range is 1 to 1000. The default is 15.
- **timeout** { **max** | **min** }—Specifies maximum and minimum retransmission intervals (in seconds) for resending control packets. Range is 1 to 8. The default maximum interval is 8; the default minimum interval is 1.

Step 6 **hello-interval** *interval*

Example:

```
RP/0/RSP0/CPU0:router(config-l2tp-class)# hello-interval 10
```

Specifies the exchange interval (in seconds) used between L2TP hello packets.

- Valid values for the *interval* argument range from 0 to 1000. The default value is 60.

Step 7 **digest** { **check disable** | **hash** { **MD5** | **SHA1** }] | **secret** { **0** | **7** } *password*]

Example:

```
RP/0/RSP0/CPU0:router(config-l2tp-class)# digest hash MD5
```

Enables L2TPv3 control-channel authentication or integrity checking.

- **secret**—Enables L2TPv3 control-channel authentication.

Note If the **digest** command is issued without the **secret** keyword option, L2TPv3 integrity checking is enabled.

- {**0** | **7**}—Specifies the input format of the shared secret. The default value is **0**.

- **0**—Specifies that a plain-text secret is entered.
- **7**—Specifies that an encrypted secret is entered.

- *password*—Defines the shared secret between peer routers. The value entered for the *password* argument must be in the format that matches the input format specified by the {**0** | **7**} keyword option.
- **hash** { **MD5** | **SHA1**}—Specifies the hash function to be used in per-message digest calculations.
    - **MD5**—Specifies HMAC-MD5 hashing (default value).
    - **SHA1**—Specifies HMAC-SHA-1 hashing.

Step 8 **hidden**

Example:

```
RP/0/RSP0/CPU0:router(config-l2tp-class)# hidden
```

Enables AVP hiding when sending control messages to an L2TPv3 peer.

Configuring L2VPN Single Segment Pseudowire

To configure single segment pseudowire in the network, do the following:

1. (Optional) Configuring the related L2VPN Global Parameters. See Configuring L2VPN Global Parameters

   This procedure is used to overwrite the default BGP Route Distinguisher (RD) auto-generated value and also the Autonomous System Number (ASN) and Route Identifier (RID) of BGP.

2. Configuring L2VPN VPWS SS-PW

3. Configuring L2VPN MS-PW Address Family Under BGP

   The address family is configured under BGP to exchange the dynamic pseudowire routes.

Configuring L2VPN Global Parameters

Perform this task to configure L2VPN global parameters.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **router-id** *router-id*
4. **pw-routing**
5. **global-id** *global-id*
6. **bgp**
7. **rd** *route-distinguisher*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 3 **router-id** *router-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router 2.2.2.2
```

Specifies the router ID.

Step 4 **pw-routing**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-routing
```

Enables pseudowire routing capabilities and enters the pseudowire routing configuration submode

Step 5 **global-id** *global-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwr)# global-id 1000
```

Configures the L2VPN global ID value for the router.

Step 6 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwr)# bgp
```

Enables the BGP pseudowire routing capabilities and enters the BGP configuration submode.

Step 7 **rd** *route-distinguisher*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwr-bgp)# rd 192.168.1.3:10
```

Configures the BGP route distinguisher.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2VPN VPWS SS-PW

Perform this task to configure L2VPN VPWS SS-PWs.

SUMMARY STEPS

1. **configure**
2. **interface type***interface-path-id*
3. **l2vpn**
4. **xconnect group** *group-name*
5. **p2p** *xconnect-name*
6. **interface** *type interface-path-id*
7. **neighbor routed** *global-id: prefix: ac-id* **source** *ac-id*
8. (optional) **pw-class** *class-name*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface type***interface-path-id*

Example:

```
RP/0/RSP0/CPU0:routerRP/0/RP00RSP0/CPU0:router# interface TenGigE0/1/0/12
```

Enters interface configuration mode and configures an interface.

Step 3 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 4 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group pw-he1
```

Configures a cross-connect group name using a free-format 32-character string.

Step 5 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p pw-ss
```

Enters P2P configuration submode.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/9
```

Specifies the interface type and instance.

Step 7 **neighbor routed** *global-id: prefix: ac-id* **source** *ac-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor routed 100:2.2.2.2:10 source 10
```

Enables pseudowire routing configuration submode for the p2p cross-connect.

Step 8 (optional) **pw-class** *class-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pwr)# pw-class dynamic_sspw
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

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

Configuring L2VPN MS-PW Address Family Under BGP

Perform this task to configure L2VPN MS-PW address family under BGP:

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family l2vpn mspw**
4. **neighbor** *ip-address*
5. **address-family l2vpn mspw**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```

Enters router configuration mode for the specified routing process.

Step 3 **address-family l2vpn mspw**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn mspw
```

Specifies the L2VPN address family and enters address family configuration mode.

Step 4 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.1
```

Adds the IP address of the neighbor in the specified autonomous system.

Step 5 **address-family l2vpn mspw**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# address-family l2vpn mspw
```

Specifies the L2VPN address family of the neighbor and enters address family configuration mode.

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

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

end - Prompts user to take one of these actions:

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

Verifying Single-Segment Pseudowires

To verify the connectivity in SS-PWs, use the ping **mpls pseudowire** command.

Displaying Information about the L2VPN Single-Segment Pseudowires

The show commands are used to display information about L2VPN Single Segment Pseudowires

- show bgp l2vpn mspw
- show l2vpn pwr summary
- show l2vpn xc

How to Configure EPVN-VPWS

The following steps are performed to configure EVPN-VPWS:

Configuring L2VPN EVPN Address Family Under BGP

Perform this task to configure L2VPN EVPN address family under BGP:

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family l2vpn evpn**
4. **neighbor** *ip-address*
5. **address-family l2vpn evpn**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```

Enters router configuration mode for the specified routing process.

Step 3 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
```

Specifies the L2VPN address family and enters address family configuration mode.

Step 4 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.1
```

Adds the IP address of the neighbor in the specified autonomous system.

Step 5 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# address-family l2vpn evpn
```

Specifies the L2VPN address family of the neighbor and enters address family configuration mode.

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

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

end - Prompts user to take one of these actions:

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

Configuring EVPN-VPWS

Perform this task to configure EVPN-VPWS.

Note You can configure the PWHE interface as well, using EVPN-VPWS. Refer to the Configuring Pseudowire Headend, on page 329 module for more information.

SUMMARY STEPS

1. **configure**
2. **interface type** *interface-path-id*
3. **l2vpn**
4. **xconnect group** *group-name*
5. **p2p** *xconnect-name*
6. **interface** *type interface-path-id*
7. **neighbor evpn evi** *vpn-id***target** *ac-id*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE0/1/0/12
```

Enters interface configuration mode and configures an interface.

Step 3 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 4 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group xc1
```

Configures a cross-connect group name using a free-format 32-character string.

Step 5 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p pw-ss
```

Enters P2P configuration submode.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/9
```

Specifies the interface type and instance.

Step 7 **neighbor evpn evi** *vpn-id***target** *ac-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 12
```

Enables EVPN-VPWS endpoint on the p2p cross-connect.

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

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

end - Prompts user to take one of these actions:

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

Configuring an Access Pseudowire using EVPN-VPWS

Bridge-domain can configure an access pseudowire using EVPN-VPWS:

SUMMARY STEPS

1. **configure**
2. **interface type** *interface-path-id*
3. **l2vpn**
4. **bridge group** *bridge-group-name*
5. **bridge-domain** *bridge-domain-name*
6. **neighbor evpn evi** *vpn-id* **target** *ac-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters the Global Configuration mode.
Step 2	**interface type** *interface-path-id* **Example:** `RP/0/RSP0/CPU0:routerRP/0/RP00RSP0/CPU0:router# interface TenGigE0/1/0/12`	Enters interface configuration mode and configures an interface.
Step 3	**l2vpn** **Example:** `RP/0/RSP0/CPU0:router(config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 4	**bridge group** *bridge-group-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group access-pw`	Creates a bridge group that can contain bridge domains, and then assigns network interfaces to the bridge domain.
Step 5	**bridge-domain** *bridge-domain-name* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1`	Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.
Step 6	**neighbor evpn evi** *vpn-id* **target** *ac-id* **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor evpn evi 100 target 12`	Enables EVPN-VPWS endpoint on the p2p cross-connect.

Example

Configuration Examples for Point to Point Layer 2 Services

This section includes these configuration examples:

L2VPN Interface Configuration: Example

The following example shows how to configure an L2VPN interface :

```
configure
 interface GigabitEthernet0/0/0/0.1 l2transport
 encapsulation dot1q 1
 rewrite ingress tag pop 1 symmetric
 end
```

Local Switching Configuration: Example

This example shows how to configure Layer 2 local switching:

```
configure
 l2vpn
  xconnect group examples
  p2p example1
  interface TenGigE0/7/0/6.5
  interface GigabitEthernet0/4/0/30
 commit
 end

show l2vpn xconnect group examples
Legend: ST = State, UP = Up, DN = Down, AD = Admin Down, UR = Unresolved,
        SB = Standby, SR = Standby Ready

XConnect                   Segment 1                   Segment 2
Group      Name       ST   Description            ST   Description            ST
examples   example1   UP   Te0/7/0/6.5            UP   Gi0/4/0/30             UP
```

Local Connection Redundancy Configuration: Example

The following example shows how to configure the LCR on PoA1:

```
! LCR - CE1
  group 107
   mlacp node 1
   mlacp system mac 0001.0001.0107
   mlacp system priority 107
    member
    neighbor 200.0.2.1
   !
   ! LCR - CE2
  group 207
   mlacp node 1
   mlacp system mac 0001.0001.0207
   mlacp system priority 207
    member
    neighbor 200.0.2.1
   !

interface Bundle-Ether107
 description CE5 - LCR
 mlacp iccp-group 107
 mlacp port-priority 10
 no shut

interface Bundle-Ether207
 description CE6 - LCR
 mlacp iccp-group 207
 mlacp port-priority 10
 no shut

interface bundle-e107.1 l2t
  description CE5 - LCR
  encap dot1q 107 second 1
  rewrite ingress tag pop 2 symmetric

interface bundle-e207.1 l2t
  description CE2 - LCR
  encap dot1q 207 second 1
  rewrite ingress tag pop 2 symmetric

interface bundle-e307.1 l2t
  description PE2 - LCR
  encap dot1q 1
  rewrite ingress tag pop 1 symmetric

l2vpn
 xconnect group lcr-scale
  p2p lcr-1
   interface bundle-e107.1
   interface bundle-e207.1
   backup interface bundle-e307.1
```

Point-to-Point Cross-connect Configuration: Examples

This section includes configuration examples for both static and dynamic p2p cross-connects.

Static Configuration

This example shows how to configure a static point-to-point cross-connect:

```
configure
  l2vpn
   xconnect group vlan_grp_1
    p2p vlan1
    interface GigabitEthernet0/0/0/0.1
    neighbor 102.2.12.1 2 pw-id 1
  mpls static label local 699 remote 890
commit2000
```

Dynamic Configuration

This example shows how to configure a dynamic point-to-point cross-connect:

```
configure
  l2vpn
   xconnect group vlan_grp_1
    p2p vlan1
    interface TenGigE 0/0/0/0.1
    neighbor 2.2.1.1 pw-id 1
commit
```

Inter-AS: Example

This example shows how to set up an AC to AC cross-connect from AC1 to AC2:

```
router-id Loopback0
```

```
interface Loopback0
ipv4 address 10.0.0.5 255.255.255.255
!
interface GigabitEthernet0/1/0/0.1 l2transport
encapsulation dot1q 1
!
!
interface GigabitEthernet0/0/0/3
ipv4 address 10.45.0.5 255.255.255.0
keepalive disable
!
interface GigabitEthernet0/0/0/4
ipv4 address 10.5.0.5 255.255.255.0
keepalive disable
!
router ospf 100
log adjacency changes detail
area 0
interface Loopback0
!
interface GigabitEthernet0/0/0/3
!
interface GigabitEthernet0/0/0/4
!
!
!
router bgp 100
address-family ipv4 unicast
allocate-label all
!
neighbor 10.2.0.5
remote-as 100
update-source Loopback0
address-family ipv4 unicast
!
address-family ipv4 labeled-unicast
!
!
!
l2vpn
xconnect group cisco
p2p cisco1
interface GigabitEthernet0/1/0/0.1
neighbor 10.0.1.5 pw-id 101
!
p2p cisco2
interface GigabitEthernet0/1/0/0.2
neighbor 10.0.1.5 pw-id 102
!
p2p cisco3
interface GigabitEthernet0/1/0/0.3
neighbor 10.0.1.5 pw-id 103
!
p2p cisco4
interface GigabitEthernet0/1/0/0.4
neighbor 10.0.1.5 pw-id 104
!
p2p cisco5
interface GigabitEthernet0/1/0/0.5
neighbor 10.0.1.5 pw-id 105
!
p2p cisco6
interface GigabitEthernet0/1/0/0.6
neighbor 10.0.1.5 pw-id 106
```

```
!
p2p cisco7
interface GigabitEthernet0/1/0/0.7
neighbor 10.0.1.5 pw-id 107
!
p2p cisco8
interface GigabitEthernet0/1/0/0.8
neighbor 10.0.1.5 pw-id 108
!
p2p cisco9
interface GigabitEthernet0/1/0/0.9
neighbor 10.0.1.5 pw-id 109
!
p2p cisco10
interface GigabitEthernet0/1/0/0.10
neighbor 10.0.1.5 pw-id 110
!
!
!
mpls ldp
router-id Loopback0
log
neighbor
!
interface GigabitEthernet0/0/0/3
!
interface GigabitEthernet0/0/0/4
!
!
end
```

L2VPN Quality of Service: Example

This example shows how to attach a service-policy to an L2 interface in port mode:

```
configure
  interface GigabitEthernet 0/0/0/0
  l2transport
  service-policy input pmap_1
commit
```

Pseudowires: Examples

The examples include these devices and connections:

- T-PE1 node has:
  - Cross-connect with an AC interface (facing CE1)
  - Pseudowire to S-PE1 node
  - IP address 209.165.200.225
- T-PE2 node
  - Cross-connect with an AC interface (facing CE2)
  - Pseudowire to S-PE1 node
  - IP address 209.165.200.254
- S-PE1 node
  - Multisegment pseudowire cross-connect with a pseudowire segment to T-PE1 node

- Pseudowire segment to T-PE2 node
- IP address 209.165.202.158

Configuring Dynamic Pseudowires at T-PE1 Node: Example

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

Configuring Dynamic Pseudowires at S-PE1 Node: Example

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

Configuring Dynamic Pseudowires at T-PE2 Node: Example

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

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

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

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

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

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

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

```
RP/0/RSP0/CPU0:T-PE2# configure
RP/0/RSP0/CPU0:T-PE2(config)# l2vpn
RP/0/RSP0/CPU0:T-PE2(config-l2vpn)# pw-class dynamic_mpls
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc-encap-mpls)# protocol ldp
RP/0/RSP0/CPU0:T-PE2(config-l2vpn-pwc-encap-mpls)# control-word disable
RP/0/RSP0/CPU0:S-PE1(config-l2vpn-pwc-encap-mpls)# preferred-path interface tunnel-te 2000
```

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

Configuring Static Pseudowires at T-PE1 Node: Example

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

Configuring Static Pseudowires at S-PE1 Node: Example

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

Configuring Static Pseudowires at T-PE2 Node: Example

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

Preferred Path: Example

This example shows how to configure preferred tunnel path:

```
configure
l2vpn
pw-class path1
encapsulation mpls
preferred-path interface  tunnel tp 50 fallback disable
```

MPLS Transport Profile: Example

This section provides examples for:

- Configuring Preferred Tunnel Path: Example
- Configuring PW Status OAM: Example

Configuring Preferred Tunnel Path: Example

This sample configuration shows how to configure preferred tunnel path:

```
l2vpn
 pw-class foo
  encapsulation mpls
  preferred-path interface tunnel-tp 100 fallback disable
 commit
```

Configuring PW Status OAM: Example

This sample configuration shows how to configure PW status OAM functionality:

```
l2vpn
 pw-oam refresh transmit 100
 commit
```

Viewing Pseudowire Status: Example

show l2vpn xconnect

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

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

show l2vpn xconnect detail

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

    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing PW Switching TLVs (Label Mapping message):
      Local IP Address: 80.80.80.80, Remote IP address: 90.90.90.90, PW ID: 300
      Description: S-PE1 MS-PW between 70.70.70.70 and 90.90.90.90
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Statistics:
      packet totals: receive 0
      byte totals: receive 0
    Create time: 04/04/2008 23:18:24 (00:01:24 ago)
    Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
  PW: neighbor 90.90.90.90, PW ID 300, state is up ( established )
    PW class not set
    Encapsulation MPLS, protocol LDP
    PW type Ethernet VLAN, control word enabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    PW Status TLV in use
        MPLS         Local                          Remote
      ------------ ------------------------------ ------------------------------
      Label        16004                          16006
      Group ID     0x2000800                      0x2000200
      Interface    GigabitEthernet0/1/0/0.3       GigabitEthernet0/1/0/2.2
      MTU          1500                           1500
      Control word enabled                        enabled
      PW type      Ethernet VLAN                  Ethernet VLAN
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x5                            0x7
                   (control word)                 (control word)
                                                  (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ ------------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing PW Switching TLVs (Label Mapping message):
      Local IP Address: 80.80.80.80, Remote IP address: 70.70.70.70, PW ID: 100
      Description: S-PE1 MS-PW between 70.70.70.70 and 90.90.90.90
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Statistics:
      packet totals: receive 0
      byte totals: receive 0
    Create time: 04/04/2008 23:18:24 (00:01:24 ago)
    Last time status changed: 04/04/2008 23:19:30 (00:00:18 ago)
```

Configuring Any Transport over MPLS: Example

This example shows you how to configure Any Transport over MPLS (AToM):

```
config
l2vpn
 xconnect group test
 p2p test
 interface POS 0/1/0/0.1
  neighbor 10.1.1.1 pw-id 100
```

Configuring AToM IP Interworking: Example

This example shows you how to configure IP interworking:

```
config
l2vpn
 xconnect group test
 p2p test
 interworking ipv4
```

Configuring PPP IP Interworking: Example

This example shows you how to configure PPP IP interworking:

```
interface Serial0/2/1/0/1/1/1:0
 encapsulation ppp
l2transport
 !
!
interface Serial0/0/0/0/2/1/1:0
encapsulation ppp
 l2transport
 !
!

!! Local Switching Configuration
l2vpn
xconnect group ppp_ip_ls
  p2p 1
   interface Serial0/2/1/0/1/1/1:0
   interface GigabitEthernet0/0/0/1.1
   interworking ipv4
  !

!! PW Configuration
l2vpn
 xconnect group ppp_ip_iw
  p2p 1
   interface Serial0/0/0/0/2/1/1:0
   neighbor 120.120.120.120 pw-id 3
    pw-class class1
   !
   interworking ipv4
```

Configuring cHDLC IP Interworking: Example

This example shows you how to configure cHDLC IP interworking:

```
interface Serial0/2/1/0/1/1/2:0
l2transport

interface Serial0/0/0/0/2/1/2:0
l2transport


!! Local Switching Configuration
```

```
l2vpn
xconnect group ppp_ip_ls
  p2p 1
   interface Serial0/2/1/0/1/1/2:0
   interface GigabitEthernet0/0/0/2.1
   interworking ipv4
  !

!! PW Configuration
l2vpn
 xconnect group ppp_ip_iw
  p2p 1
   interface Serial0/0/0/0/2/1/2:0
   neighbor 120.120.120.120 pw-id 3
    pw-class class1
   !
   interworking ipv4
```

Configuring MLPPP IP Interworking: Example

This example shows you how to configure MLPPP IP interworking:

```
interface Multilink0/2/1/0/1
 multilink
l2transport
!

interface Multilink0/2/1/0/51
Multilink
l2transport


!! Local Switching Configuration
l2vpn
 xconnect group mlppp_ip_ls
  p2p 1
   interface Multilink0/2/1/0/1
   interface GigabitEthernet0/0/0/1.151
   interworking ipv4
!
!! PW Configuration
l2vpn
xconnect group mlppp_ip_iw
  p2p 151
   interface Multilink0/2/1/0/51
   neighbor 140.140.140.140 pw-id 151
    pw-class test
   !
   interworking ipv4
  !
```

Configuring Circuit Emulation Over Packet Switched Network: Example

This example shows you how to configure Circuit Emulation Over Packet Switched Network:

Adding CEM Attachment Circuit to PW

```
l2vpn
 xconnect group gr1
  p2p p1
   interface CEM 0/0/0/0:10
   neighbor 3.3.3.3 pw-id 11
   !
  !
```

Associating Pseudowire Class

```
l2vpn
 pw-class class-cem
  encapsulation mpls
   protocol ldp
  !
 !
xconnect group gr1
  p2p p1
   interface CEM0/0/0/0:20
   neighbor 1.2.3.4 pw-id 11
    pw-class class-cem
   !
```

Enabling Pseudowire Status

```
l2vpn
  pw-status
  commit
```

Disabling Pseudowire Status

```
l2vpn
  pw-status disable
  commit
```

Configuring Backup Pseudowire

```
l2vpn
 pw-status
 pw-class class-cem
  encapsulation mpls
   protocol ldp
  !
 !
 xconnect group gr1
  p2p p1
   interface CEM0/0/0/0:20
   neighbor 1.2.3.4 pw-id 11
    pw-class class-cem
    backup neighbor 9.9.9.9 pw-id 1221
     pw-class class-cem
    !
   !
```

Configuring L2VPN Nonstop Routing: Example

This example shows how to configure L2VPN Nonstop Routing.

```
config
l2vpn
 nsr
 logging nsr
```

Enabling Pseudowire Grouping: Example

This example shows how to enable pseudowire grouping.

```
config
l2vpn
 pw-grouping
```

Configuring L2TPv3 over IPv6 Tunnels: Example

This section provides examples for:

Configuring Neighbor AFI for Pseudowire: Example

To support IPv6 pseudowire neighbors, an AFI needs to be configured as follows:

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
```

Configuring L2TPv3 encapsulation and protocol: Example

For L2TPv3 tunnels, the encapsulation and protocol has to be set to l2tpv3.

Note The default encapsulation and protocol is MPLS.

```
l2vpn
pw-class ts
encapsulation l2tpv3
protocol l2tpv3
```

Configuring Source IPv6 address for L2TPv3 over IPv6 tunnel: Example

This example shows how to configure source IPv6 address for the L2TPv3 over IPv6 tunnel:

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
source 1111:2222::abcd
```

Configuring Local and Remote Sessions: Example

For L2TPv3 over IPv6 tunnels, the local and remote session ID is configured under the pseudowire; however, this configuration is optional.

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
l2tp static local session 1
l2tp static remote session 1
```

Configuring Local and Remote Cookies: Example

For L2TPv3 over IPv6 tunnels, the local and remote cookies are configured under the pseudowire. Support has been extended for cookie roll-over that provides the ability to configure a secondary local cookie. This example shows how to configure a cookie with size 0:

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
l2tp static local cookie size 0
l2tp static remote cookie size 0
```

This example shows how to configure a cookie with size 4:

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
l2tp static local cookie size 4 value <0x0-0xffffffff>
l2tp static remote cookie size 4 value <0x0-0xffffffff>
```

This example shows how to configure a cookie with size 8 (the lower 4 bytes are entered first; followed by the higher 4 bytes):

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
l2tp static local cookie size 8 value <0x0-0xffffffff> <0x0-0xffffffff>
l2tp static remote cookie size 8 value <0x0-0xffffffff> <0x0-0xffffffff>
```

To support cookie roll-over on L2TPv3 over IPv6 tunnels, configure a secondary local cookie. The local cookie secondary command specifies the secondary cookie value on the local router.

Note The primary and secondary cookies must be of the same size. Primary or secondary local cookies must match the cookie value being received from the remote end, otherwise, packets are dropped.

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
l2tp static local cookie secondary size 8 value <0x0-0xffffffff> <0x0-0xffffffff>
```

Enabling L2TP Static Submode: Example

This example shows you how to enable the L2TP static submode:

```
l2vpn
xconnect group g1
p2p xc3
interface GigabitEthernet0/0/0/4.2
neighbor ipv6 1111:2222::cdef pw-id 1
l2tp static
local cookie <>
```

Enabling TOS Reflection in the L2TPv3 Header: Example

For L2TPv3 over IPv6 tunnels, configurations are supported for each pseudowire class to enable type of service (TOS) reflection, or to set a specific TOS value in the L2TPv3 header.

Note By default, the TOS is copied over from the class of service (COS) fields of the VLAN header. If the underlying packet is not an IPv4 or IPv6 packet, the COS fields are copied from the VLAN header, even if a TOS reflection is configured.

This example shows how to configure a TOS reflection in the L2TPv3 header:

```
l2vpn
pw-class ts
encapsulation l2tpv3
protocol l2tpv3
tos reflect

This example shows how to set a TOS value in the L2TPv3 header:
l2vpn
pw-class ts
encapsulation l2tpv3
protocol l2tpv3
tos value 64
```

Configuring TTL for L2TPv3 over IPv6 Tunnels: Example

For L2TPv3 over IPv6 tunnels, TTL configuration is supported in the pseudowire class.

```
l2vpn
pw-class ts
encapsulation l2tpv3
protocol l2tpv3
ttl <1-255>
```

Configuring Traffic Mirroring over L2TPv3 over IPv6 Tunnels: Example

This example associates an EFP to a monitor-session:

```
interface GigabitEthernet0/0/0/4.2 l2transport
monitor-session customer-foo
```

Layer 2 SPAN is supported on L3 interfaces; however, the Layer 2 frame is mirrored:

```
interface GigabitEthernet0/0/0/4.2
ipv6 address <>
monitor-session customer-foo
```

SPAN is also supported on main interfaces.

```
interface GigabitEthernet0/4/0/3
l2transport
monitor-session customer-foo
```

This example creates the monitor-session globally:

```
monitor-session customer-foo
destination pseudowire
```

This example creates a cross-connect between the monitor-session and a L2TPv3 over IPv6 tunnel:

```
l2vpn
xconnect group span
p2p span-foo
monitor-session customer-foo
neighbor ipv6 1111:3333::cdef pw-id 1001
pw-class ts
source 1111:3333::abcd
l2tp static local cookie size 8 value 0xabcd 0x1234
l2tp static remote cookie size 8 value 0xcdef 0x5678
```

For more information on:

- L2TPv3 over IPv6 tunnel concepts, see L2TPv3 over IPv6
- Configuration procedures, see Configuring L2TPv3 over IPv6 Tunnels

Configuring L2TPv3 over IPv4 Tunnels: Example

This section provides examples for:

Configuring a Dynamic L2TPv3 Pseudowire

Perform this task to configure a dynamic L2TPv3 pseudowire connecting to a remote IPv4 peer.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *name*
4. **p2p** *name*
5. **interface** *type interface-path-id*
6. **neighbor ipv4** *ip-address* **pw-id** *number*
7. **pw-class** *pw-class-name*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enter L2VPN configure submode.

Step 3 **xconnect group** *name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group L2TPV3_V4_XC_GRP
```

Enter a name for the cross-connect group.

Step 4 **p2p** *name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p L2TPV3_P2P_1
```

Enters p2p configuration submode to configure point-to-point cross-connects.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/2/0/0/0.1
```

Specifies the interface type ID. The choices are:

- GigabitEthernet
- TenGigE

Step 6 **neighbor ipv4** *ip-address* **pw-id** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor ipv4 26.26.26.26 pw-id 100
```

Configures a pseudowire for a cross-connect.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class L2TPV3_V4_CLASS
```

Enters pseudowire class submode to define a name for the cross-connect.

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

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

end - Prompts user to take one of these actions:

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

Configuring L2TPv3 Encapsulation and Protocol: Example

This example shows how to set the encapsulation and protocol for L2TPv3 tunnels:

```
configure
l2vpn
```

```
  pw-class L2TPV3_V4_CLASS
   encapsulation l2tpv3
    protocol l2tpv3 class L2TP-CLASS
    dfbit set
    ipv4 source 25.25.25.25
    cookie size 4
  !
!
```

Configuring L2TP Control-Channel Parameters: Example

The following example shows a typical L2TPv3 control-channel configuration:

```
configure
l2tp-class L2TP-CLASS
 authentication
 retransmit retries 5
 retransmit initial retries 10
 retransmit initial timeout max 5
 retransmit timeout max 6
 hidden
 password 7 1511021F07257A767B
 hello-interval 10
 digest hash MD5
!
```

Configuration Examples for EVPN-VPWS

Configuring EVPN-VPWS: Example

The following example shows how to configure EVPN-VPWS service.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group pw-he1
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p pw-ss
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface gigabitethernet 0/1/0/9
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 12 source 10
```

The following example shows how to configure EVPN-VPWS into PWHE interface.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group xg1
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p pwhe1
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface PW-Ether 1
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 20 source 20
```

Configuring an Access PW using EVPN-VPWS: Example

The following example shows how a bridge-domain can configure an access pseudowire using EVPN-VPWS.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor evpn evi 1 target 100
```

CHAPTER 6

Implementing Multipoint Layer 2 Services

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

Note VPLS supports Layer 2 VPN technology and provides transparent multipoint Layer 2 connectivity for customers. This approach enables service providers to host a multitude of new services such as broadcast TV and Layer 2 VPNs.

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

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

Feature History for Implementing Multipoint Layer 2 Services

Release	Modification
Release 3.7.2	This feature was introduced.
Release 3.9.0	These features were added: • Blocking unknown unicast flooding. • Disabling MAC flush. • Multiple Spanning Tree Access Gateway • Scale enhancements were introduced. See Table 1 for more information on scale enhancements.
Release 3.9.1	Support for VPLS with BGP Autodiscovery and LDP Signaling was added.
Release 4.0.1	Support was added for the following features: • Dynamic ARP Inspection • IP SourceGuard • MAC Address Security

Release	Modification
Release 4.1.0	Support was added for these VPLS features on the ASR 9000 SIP-700 line card: • MAC learning and forwarding. • MAC address aging support • MAC Limiting • Split Horizon Group • MAC address Withdrawal • Flooding of unknown unicast, broadcast and multicast packets • Access pseudowire • H-VPLS PW-access • PW redundancy Support was added for the G.8032 Ethernet Ring Protection feature.
Release 4.2.1	Support was added for Flow Aware Transport (FAT) Pseudowire feature.
Release 4.3.0	Support was added for these features: • Pseudowire Headend (PWHE) • Scale enhancements on ASR 9000 Enhanced Ethernet line card: • Support for 128000 pseudowires within VPWS and VPLS • Support for 128000 pseudowires across VPLS and VPWS instances • Support for upto 512 pseudowires in a bridge • Support for 128000 bundle attachment circuits • Support for 128000 VLANs • L2VPN over GRE
Release 4.3.1	Support was added for: • VC type 4 in VPLS with BGP Autodiscovery • IPv6 support for PWHE

Release	Modification
Release 5.1.0	Support was added for Multipoint Layer 2 Services Label Switched Multicast feature.
Release 5.1.1	Support was added for: • Pseudowire Headend PW-Ether sub-interfaces (VC Type 5) and Pseudowire Headend PW-IW interworking interfaces (VC Type 11) • LFA over Pseudowire Headend.
Release 6.1.2	Support was added for: • Service Path Preference for L2VPN • L2VPN Route Policy
Release 6.6.1	The MAC security recovery feature was introduced.
Release 7.4.1	From this release onwards, we'll deprecate the Multipoint Layer 2 Services Label Switched Multicast feature.

Prerequisites for Implementing Multipoint Layer 2 Services

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

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

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

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

- Configure a loopback interface to originate and terminate Layer 2 traffic. Make sure that the PE routers can access the other router's loopback interface.

Note The loopback interface is not needed in all cases. For example, tunnel selection does not need a loopback interface when Multipoint Layer 2 Services are directly mapped to a TE tunnel.

- Configure MPLS and Label Distribution Protocol (LDP) in the core so that a label switched path (LSP) exists between the PE routers.

Information About Implementing Multipoint Layer 2 Services

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

Multipoint Layer 2 Services Overview

Multipoint Layer 2 Services enable geographically separated local-area network (LAN) segments to be interconnected as a single bridged domain over an MPLS network. The full functions of the traditional LAN such as MAC address learning, aging, and switching are emulated across all the remotely connected LAN segments that are part of a single bridged domain.

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

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

Bridge Domain

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

By default, split horizon is enabled for pseudowires under the same VFI. However, in the default configuration, split horizon is not enabled on the attachment circuits (interfaces or pseudowires).

Flood Optimization

A Cisco ASR 9000 Series Router, while bridging traffic in a bridge domain, minimizes the amount of traffic that floods unnecessarily. The Flood Optimization feature accomplishes this functionality. However, in certain failure recovery scenarios, extra flooding is actually desirable in order to prevent traffic loss. Traffic loss occurs during a temporary interval when one of the bridge port links becomes inactive, and a standby link replaces it.

In some configurations, optimizations to minimize traffic flooding is achieved at the expense of traffic loss during the short interval in which one of the bridge's links fails, and a standby link replaces it. Therefore, Flood Optimization can be configured in different modes to specify a particular flooding behavior suitable for your configuration.

These flood optimization modes can be configured:

Bandwidth Optimization Mode

Flooded traffic is sent only to the line cards on which a bridge port or pseudowire that is attached to the bridge domain resides. This is the default mode.

Convergence Mode

Flooded traffic is sent to all line cards in the system. Traffic is flooded regardless of whether they have a bridge port or a pseudowire that is attached to the bridge domain. If there are multiple Equal Cost MPLS Paths (ECMPs) attached to that bridge domain, traffic is flooded to all ECMPs.

The purpose of Convergence Mode is to ensure that an absolute minimum amount of traffic is lost during the short interval of a bridge link change due to a failure.

TE FRR Optimized Mode

The Traffic Engineering Fast Reroute (TE FRR) Optimized Mode is similar to the Bandwidth Optimized Mode, except for the flooding behavior with respect to any TE FRR pseudowires attached to the bridge domain. In TE FRR Optimized Mode, traffic is flooded to both the primary and backup FRR interfaces. This mode is used to minimize traffic loss during an FRR failover, thus ensuring that the bridge traffic complies with the FRR recovery time constraints.

Dynamic ARP Inspection

Dynamic ARP Inspection (DAI) is a method of providing protection against address resolution protocol (ARP) spoofing attacks. It intercepts, logs, and discards ARP packets with invalid IP-to-MAC address bindings. This capability protects the network from certain man-in-the-middle attacks. The DAI feature is disabled by default.

ARP enables IP communication within a Layer 2 broadcast domain by mapping an IP address to a MAC address. Spoofing attacks occur because ARP allows a response from a host even when an ARP request is not actually received. After an attack occurs, all traffic, from the device under attack, first flows through the attacker's system, and then to the router, switch, or the host. An ARP spoofing attack affects the devices connected to your Layer 2 network by sending false information to the ARP caches of the devices connected to the subnet. The sending of false information to an ARP cache is known as ARP cache poisoning.

The Dynamic ARP Inspection feature ensures that only valid ARP requests and responses are relayed. There are two types of ARP inspection:

- Mandatory inspection—The sender’s MAC address, IPv4 address, receiving bridge port XID and bridge are checked.
- Optional inspection—The following items are validated:
  - Source MAC: The sender’s and source MACs are checked. The check is performed on all ARP or RARP packets.
  - Destination MAC: The target and destination MACs are checked. The check is performed on all Reply or Reply Reverse packets.
  - IPv4 Address: For ARP requests, a check is performed to verify if the sender’s IPv4 address is 0.0.0.0, a multicast address or a broadcast address. For ARP Reply and ARP Reply Reverse, a check is performed to verify if the target IPv4 address is 0.0.0.0, a multicast address or a broadcast address. This check is performed on Request, Reply and Reply Reverse packets.

Note The DAI feature is supported on attachment circuits and EFPs. Currently, the DAI feature is not supported on pseudowires.

IP Source Guard

IP source guard (IPSG) is a security feature that filters traffic based on the DHCP snooping binding database and on manually configured IP source bindings in order to restrict IP traffic on non-routed Layer 2 interfaces.

The IPSG feature provides source IP address filtering on a Layer 2 port, to prevent a malicious hosts from manipulating a legitimate host by assuming the legitimate host's IP address. This feature uses dynamic DHCP snooping and static IP source binding to match IP addresses to hosts.

Initially, all IP traffic, except for DHCP packets, on the EFP configured for IPSG is blocked. After a client receives an IP address from the DHCP server, or after static IP source binding is configured by the administrator, all traffic with that IP source address is permitted from that client. Traffic from other hosts is denied. This filtering limits a host's ability to attack the network by claiming a neighbor host's IP address.

Note The IPSG feature is supported on attachment circuits and EFPs. Currently, the IPSG feature is not supported on pseudowires.

Pseudowires

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

The following scale enhancements are applicable to ASR 9000 Enhanced Ethernet line card:

- Support for 128000 pseudowires within VPWS and VPLS
- Support for 128000 pseudowires across VPLS and VPWS instances
- Support for upto 512 pseudowires in a bridge

Note This scale enhancement is supported in hardware configurations where RSP3 and ASR 9000 Enhanced Ethernet line cards are used. However, these enhancements are not applicable to the RSP2, ASR 9000 Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 line cards.

DHCP Snooping over Pseudowire

The Cisco ASR 9000 Series Routers provide the ability to perform DHCP snooping, where the DHCP server is reachable on a pseudowire. The Pseudowire is considered as a trusted interface.

The dhcp ipv4 snoop profile {*dhcp-snooping-profile1*} command is provided under the bridge domain to enable DHCP snooping on a bridge and to attach a DHCP snooping profile to the bridge.

Virtual Forwarding Instance

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

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

VPLS for an MPLS-based Provider Core

VPLS is a multipoint Layer 2 VPN technology that connects two or more customer devices using bridging techniques. A bridge domain, which is the building block for multipoint bridging, is present on each of the PE routers. The access connections to the bridge domain on a PE router are called attachment circuits. The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

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

The MPLS/IP provider core simulates a virtual bridge that connects the multiple attachment circuits on each of the PE devices together to form a single broadcast domain. This also requires all of the PE routers that are participating in a VPLS instance to form emulated virtual circuits (VCs) among them.

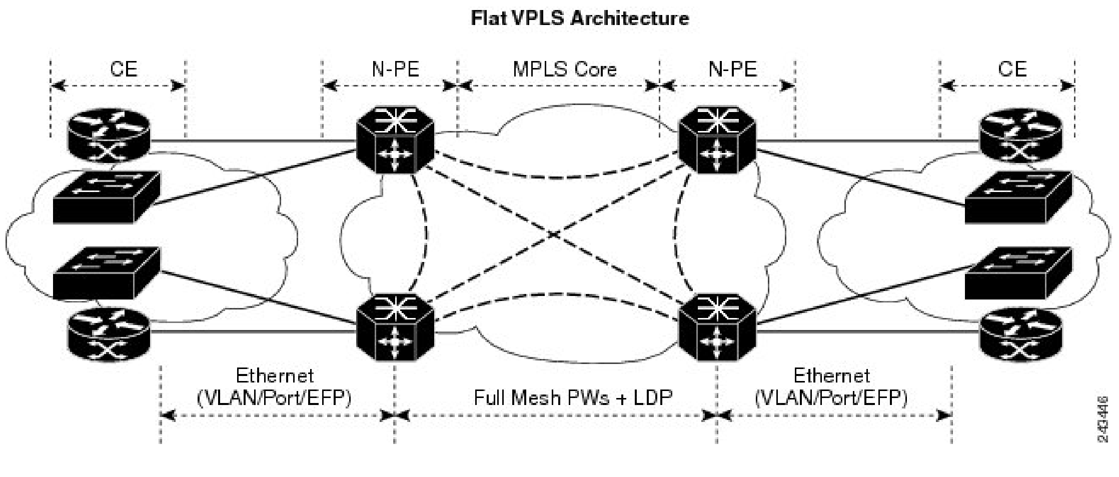
Now, the service provider network starts switching the packets within the bridged domain specific to the customer by looking at destination MAC addresses. All traffic with unknown, broadcast, and multicast destination MAC addresses is flooded to all the connected customer edge devices, which connect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

In the MPLS provider core, the VPLS pseudowire traffic can be dynamically routed over any interface that supports LDP protocol.

VPLS Architecture

The basic or flat VPLS architecture allows for the end-to-end connection between the provider edge (PE) routers to provide multipoint ethernet services. Following figure shows a flat VPLS architecture illustrating the interconnection between the network provider edge (N-PE) nodes over an IP/MPLS network.

Figure 19: Basic VPLS Architecture



The VPLS network requires the creation of a Bridge Domain (Layer 2 broadcast domain) on each of the PE routers. The VPLS provider edge device holds all the VPLS forwarding MAC tables and bridge domain information. In addition, it is responsible for all flooding broadcast frames and multicast replications.

The PEs in the VPLS architecture are connected with a full mesh of Pseudowires (PWs). A Virtual Forwarding Instance (VFI) is used to interconnect the mesh of pseudowires. A bridge domain is connected to a VFI to create a Virtual Switching Instance (VSI), that provides Ethernet multipoint bridging over a PW mesh. VPLS network links the VSIs using the MPLS pseudowires to create an emulated Ethernet Switch.

With VPLS, all customer equipment (CE) devices participating in a single VPLS instance appear to be on the same LAN and, therefore, can communicate directly with one another in a multipoint topology, without requiring a full mesh of point-to-point circuits at the CE device. A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

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

VPLS for Layer 2 Switching

VPLS technology includes the capability of configuring the Cisco ASR 9000 Series Routers to perform Layer 2 bridging. In this mode, the Cisco ASR 9000 Series Routers can be configured to operate like other Cisco switches.

These features are supported:

- Bridging IOS XR Trunk Interfaces
- Bridging on EFPs

Refer to the Configuration Examples for Multipoint Layer 2 Services section for examples on these bridging features.

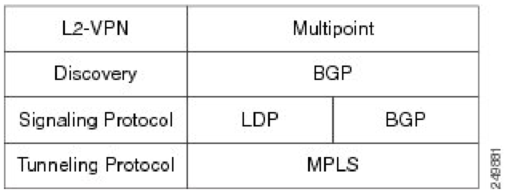
VPLS Discovery and Signaling

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

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

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

Figure 20: VPLS Autodiscovery and Signaling



L2-VPN	Multipoint	
Discovery	BGP	
Signaling Protocol	LDP	BGP
Tunneling Protocol	MPLS	

BGP-based VPLS Autodiscovery

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

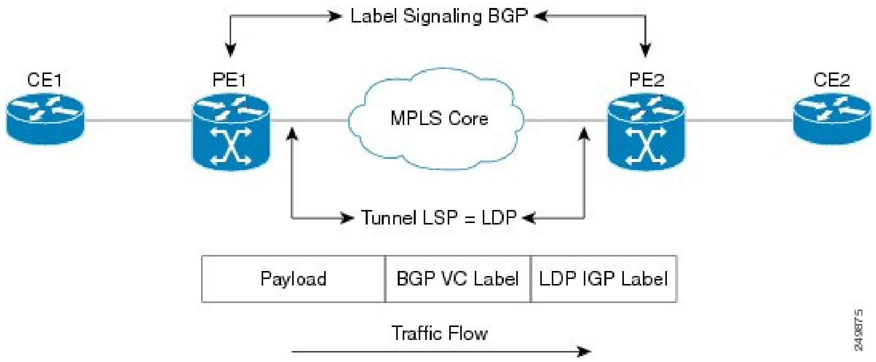
BGP-based VPLS autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS autodiscovery enables each VPLS PE router to discover the other provider edge (PE) routers that are part of the same VPLS domain. VPLS Autodiscovery also tracks when PE routers are added to or removed from the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

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

BGP Auto Discovery With BGP Signaling

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

Figure 21: Discovery and Signaling Attributes



The BGP signaling and autodiscovery scheme has the following components:

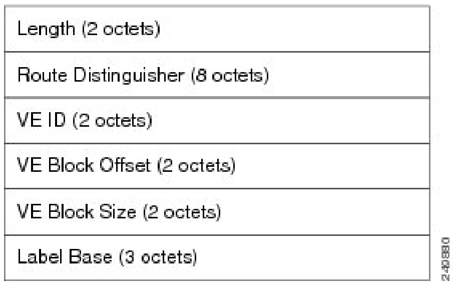
- A means for a PE to learn which remote PEs are members of a given VPLS. This process is known as autodiscovery.
- A means for a PE to learn the pseudowire label expected by a given remote PE for a given VPLS. This process is known as signaling.

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

NLRI Format for VPLS with BGP AD and Signaling

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

Figure 22: NLRI Format

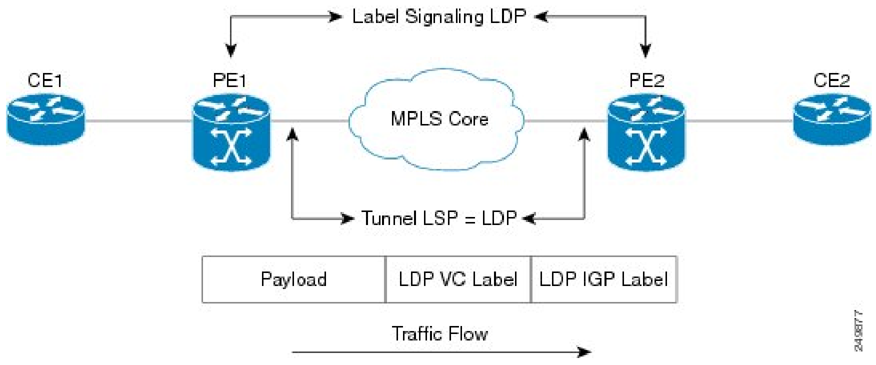


Length (2 octets)
Route Distinguisher (8 octets)
VE ID (2 octets)
VE Block Offset (2 octets)
VE Block Size (2 octets)
Label Base (3 octets)

BGP Auto Discovery With LDP Signaling

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

Figure 23: Discovery and Signaling Attributes



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

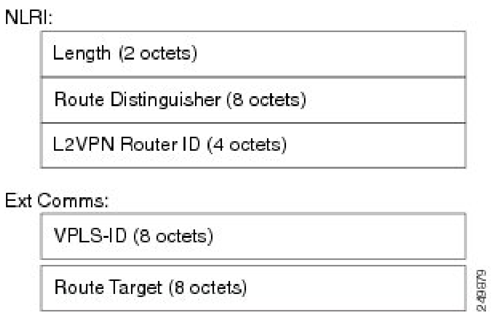
The LDP protocol is used to configure a pseudowire to all the other PE routers. FEC 129 is used for the signaling. The information carried by FEC 129 includes the VPLS ID, the Target Attachment Individual Identifier (TAII) and the Source Attachment Individual Identifier (SAII).

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

NLRI and Extended Communities

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

Figure 24: NLRI and Extended Communities



Service Path Preference for L2VPN

Service Path Preference feature (SPP) helps control transport path for L2VPN services in traffic engineering (TE) tunnels. SPP feature provides a way for services to influence path selection while forwarding in Multiprotocol Label Switching (MPLS) networks. SPP is achieved by associating a control plane policy with a forward-class. SPP is supported for per-service path selection for VPLS with BGP AD, and EVI for PBB-EVPN and EVPN.

SPP for L2VPN is implemented in two steps:

- Path selection - classify incoming traffic on a per service basis.
- Path setup - set up paths with forward-class service.

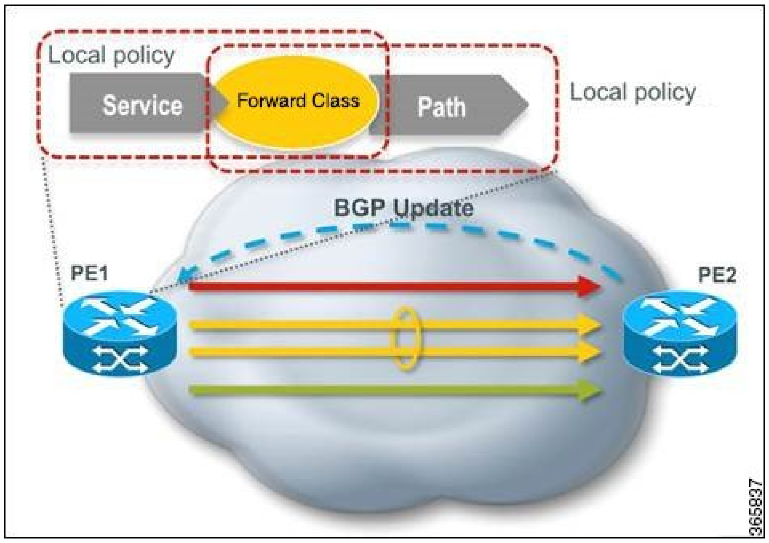
Refer to *Service Path Preference for MPLS VPN Sessions* module for more information on SPP.

Understanding How Service Path Preference Works

SPP allows services to select a path based on policies configured in the control plane.

Consider a scenario where you have two Provider Edge (PE) routers in a setup. PE1 functions as ingress node and PE2 functions as egress node.

Figure 25: Sample Service Path Preference scenario



The ingress PE (PE1) receives the routes from the customers. The local policies determine the attribute to be assigned to the customer. PE1 associates a forward-class to the prefix based on the local policies that are created based on the VFI or EVI service. The pre-configured tunnel with matching forward-class is selected for forwarding the traffic.

L2VPN Route Policy

L2VPN route-policy feature enables the export community configuration for L2VPN VPWS and L2VPN VPLS with BGP autodiscovery. BGP executes the route-policy. This functionality is similar to the route-policy support under BGP submode for L3VPN services.

The following points explain the L2VPN route-policy functionality:

1. The RPL is used to set standard community as policy action for two new attachpoints that are L2VPN export VPLS (VFI) and L2VPN export VPWS (MP2MP).
2. L2VPN configuration specifies which route-policy to use for a given bridge-domain configured with BGP autodiscovery.
3. L2VPN sends the route-policy name to BGP along with the L2VPN context.
4. BGP process inserts standard community to the L2 NLRI.

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

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

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

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

MAC Address-related Parameters

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

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

Note After you modify the MAC limit or action at the bridge domain level, ensure that you shut and unshut the bridge domain for the action to take effect. If you modify the MAC limit or action on an attachment circuit (through which traffic is passing), the attachment circuit must be shut and unshut for the action to take effect.

MAC Address Flooding

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

MAC Address-based Forwarding

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

Note Split horizon forwarding applies in this case, for example, frames that are coming in on an attachment circuit or pseudowire are sent out of the same pseudowire. The pseudowire frames, which are received on one pseudowire, are not replicated on other pseudowires in the same virtual forwarding instance (VFI).

MAC Address Source-based Learning

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

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. The updated MAC tables are propagated and programs the hardware for the router.

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

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

MAC Address Aging

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

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

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

MAC Address Limit

The MAC address limit is used to limit the number of learned MAC addresses. The bridge domain level limit is always configured and cannot be disabled. The default value of the bridge domain level limit is 4000 and can be changed in the range of 1-512000.

Configure MAC Address Limit

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

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

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

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

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

By default, syslog message is sent.

Configuration Example

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

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

Verification

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

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

MAC Address Withdrawal

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

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

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

The LDP MAC Withdrawal feature is generated due to these events:

- Attachment circuit goes down. You can remove or add the attachment circuit through the CLI.

- MAC withdrawal messages are received over a VFI pseudowire. RFC 4762 specifies that both wildcards (by means of an empty Type, Length and Value [TLV]) and a specific MAC address withdrawal. Cisco IOS XR software supports only a wildcard MAC address withdrawal.

MAC Address Security

You can configure MAC address security at the interfaces and at the bridge access ports (subinterfaces) levels. However, MAC security configured under an interface takes precedence to MAC security configured at the bridge domain level. When a MAC address is first learned, on an EFP that is configured with MAC security and then, the same MAC address is learned on another EFP, these events occur:

- the packet is dropped
- the second EFP is shutdown
- the packet is learned and the MAC from the original EFP is flushed

MAC Loop Prevention

Table 4: Feature History Table

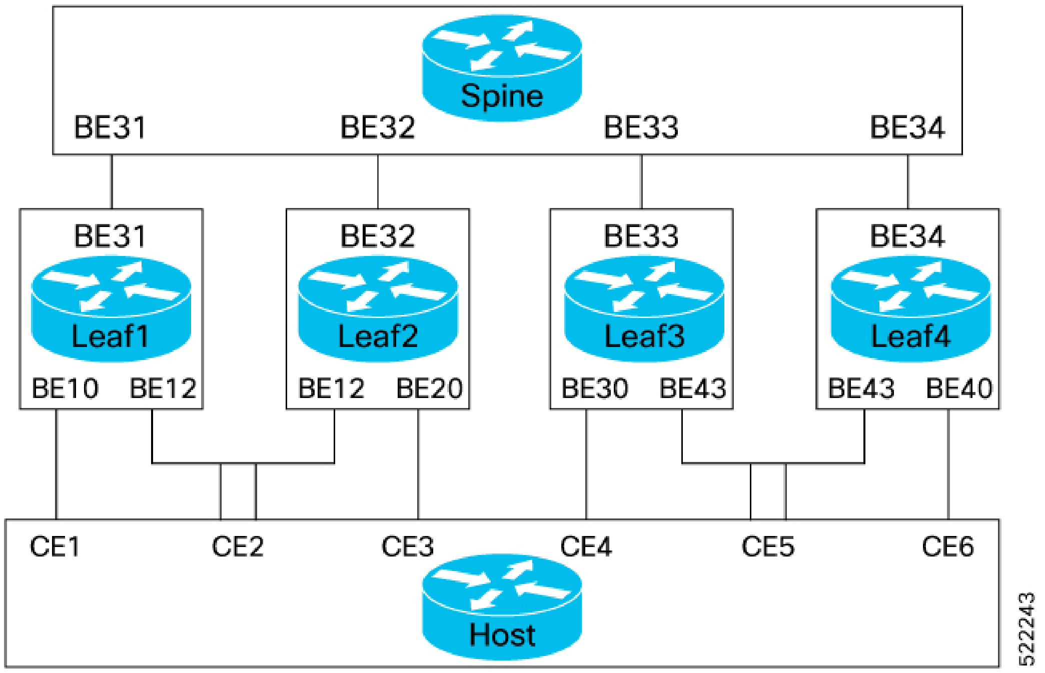
Feature Name	Release Information	Feature Description
MAC Loop Prevention	Release 7.5.2	This feature helps reduce network congestion and avoid traffic loss by shutting down a port after it reaches the configured number of MAC moves within the specified move interval. You can configure this feature at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

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

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

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

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



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

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

Configuration Example

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

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

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

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

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

Running Configuration

This section shows the MAC loop prevention running configuration.

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

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

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

Verification

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

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

MAC Security Recovery

MAC security recovery feature enables you to recover the bridge port or Ethernet flow point (EFP) that is shut down due to MAC security violation. The recovery is achieved by configuring a shutdown recovery timer, using the **mac secure shutdown-recovery-timeout** command. The bridge port or EFP that is shut down, is UP automatically after the timer expires. The MAC security recovery feature takes effect only when you have enabled *shutdown* to be the default action for violating MAC security policies.

Restrictions for MAC Security Recovery

- The MAC security recovery applies only for the EFP security.
- The shutdown recovery timer does not apply to MAC limits configured on a per-EFP level, per-bridge domain level, or both.

For more information on configuring MAC security recovery, see Configure MAC Security Recovery, on page 228section.

Configure MAC Security Recovery

Perform this task to configure MAC security recovery under the following:

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

```
/* Configure MAC security recovery under bridge domain */

Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd-1-6
Router(config-l2vpn-bg-bd)# mac
Router(config-l2vpn-bg-bd-mac)# aging
Router(config-l2vpn-bg-bd-mac-aging)# time 900
Router(config-l2vpn-bg-bd-mac-aging)# secure
Router(config-l2vpn-bg-bd-mac-secure)# action shutdown
Router(config-l2vpn-bg-bd-mac-secure)# logging
Router(config-l2vpn-bg-bd-mac-secure)# shutdown-recovery-timeout 100
Router(config-l2vpn-bg-bd-mac-secure)#    !
```

```
/* Configure MAC security recovery under bridge port or attachment circuit (AC) */

Router(config-l2vpn-bg-bd-mac-secure)# interface GigabitEthernet 0/0/0/5.11
Router(config-l2vpn-bg-bd-ac)# mac
Router(config-l2vpn-bg-bd-ac-mac)# secure
Router(config-l2vpn-bg-bd-ac-mac-secure)# action shutdown
Router(config-l2vpn-bg-bd-ac-mac-secure)# logging
Router(config-l2vpn-bg-bd-ac-mac-secure)# shutdown-recovery-timeout 600
Router(config-l2vpn-bg-bd-ac-mac-secure)# !
```

```
/* Configure MAC security recovery over access pseudowire (PW) */

Router(config-l2vpn-bg-bd-ac-mac-secure)# neighbor 64.64.64.3 pw-id 2300001
Router(config-l2vpn-bg-bd-pw)# !
Router(config-l2vpn-bg-bd-pw)# neighbor 64.64.64.4 pw-id 2400001
Router(config-l2vpn-bg-bd-pw)# mac
Router(config-l2vpn-bg-bd-pw-mac)# secure
Router(config-l2vpn-bg-bd-pw-mac-secure)# action shutdown
Router(config-l2vpn-bg-bd-pw-mac-secure)# logging
Router(config-l2vpn-bg-bd-pw-mac-secure)# shutdown-recovery-timeout 600
Router(config-l2vpn-bg-bd-pw-mac-secure)# !
```

Running Configuration for MAC Security Recovery

This example shows how to configure MAC security recovery for L2VPN under bridge domain.

```
config
l2vpn
 bridge group bg1
 bridge-domain bd-1-6
  mac
    aging
      time 900
!
  secure
   action shutdown
     logging
       shutdown-recovery-timeout 30
       !
     !
```

This example shows how to configure MAC security recovery for L2VPN under attachment circuit (AC).

```
Interface GigabitEthernet 0/0/0/5.11
mac
  secure
   action shutdown
     logging
       shutdown-recovery-timeout 600
```

This example shows how to configure MAC security recovery for L2VPN over access pseudowires.

```
neighbor 64.64.64.3 pw-id 2300001
!
neighbor 64.64.64.4 pw-id 2400001
 mac
  secure
   action shutdown
     logging
       shutdown-recovery-timeout 600
```

Verify MAC Security Recovery

This sample output verifies that the MAC security recovery is enabled under bridge domain, and also displays the value that is set for Shutdown recovery timer.

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain bd-name bd-1-6 detail
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd-1-6, id: 5, 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: 900 s, Type: inactivity
  MAC limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no, threshold: 75%
  MAC port down flush: enabled
  MAC Secure: enabled, Logging: enabled, Action: shutdown, Threshold: disabled
  MAC Secure Shutdown recovery timer : 30 sec
  Split Horizon Group: none
```

This sample output verifies that the MAC security recovery is enabled under an attachment circuit (AC), and also displays the value that is set for Shutdown recovery timer.

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain bd-name bd-1-6 detail

ACs: 1 (1 up), VFIs: 0, PWs: 2 (2 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    AC: GigabitEthernet0/0/0/5.11, state is up
      Type VLAN; Num Ranges: 1
      Outer Tag: 21
      Rewrite Tags: []
      VLAN ranges: [11, 11]
      MTU 1500; XC ID 0x1205e26; interworking none
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: e nabled
        Unknown unicast: enabled
      MAC aging time: 900 s, Type: inactivity
      MAC limit: 4000, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
      MAC Secure: enabled, Logging: enabled, Action: shutdown, Threshold: disabled
      MAC Secure Shutdown recovery timer : 600 sec
      Split Horizon Group: none
      E-Tree: Root
```

This sample output verifies that the MAC security recovery is enabled under access pseudowires, and also displays the value that is set for Shutdown recovery timer.

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain bd-name bd-1-6 detail

List of Access PWs:
    PW: neighbor 64.64.64.3, PW ID 2300001, state is up ( established )
      PW class not set, XC ID 0xa0000019
      Encapsulation MPLS, protocol LDP
      Source address 64.64.64.2
      PW type Ethernet, control word disabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
      LSP : Up
      PW Status TLV in use
        MPLS         Local                          Remote
        ------------ ------------------------------ ---------------------------
        Label        60029                          24013
        Group ID     0x5                            0x5
        Interface    Access PW                      Access PW
        MTU          1500                           1500
        Control word disabled                       disabled
        PW type      Ethernet                       Ethernet
        VCCV CV type 0x2                            0x2
                     (LSP ping verification)        (LSP ping verification)
        VCCV CC type 0x6                            0x6
                     (router alert label)           (router alert label)
                     (TTL expiry)                   (TTL expiry)
        ------------ ------------------------------ ---------------------------
      Incoming Status (PW Status TLV):
        Status code: 0x0 (Up) in Notification message
      MIB cpwVcIndex: 2684354585
      Create time: 14/08/2018 11:51:56 (04:28:11 ago)
      Last time status changed: 14/08/2018 11:51:57 (04:28:10 ago)
      MAC aging time: 900 s, Type: inactivity
      MAC limit: 4000, Action: none, Notification: syslog
      MAC limit reached: no, threshold: 75%
      MAC port down flush: enabled
     MAC Secure: enabled, Logging: enabled, Action: shutdown, Threshold: disabled
     MAC Secure Shutdown recovery timer: 30 sec
```

You can also use the **Show l2vpn forwarding interface interface-name hardware ingress detail location location-id** command to verify the values that are configured for the Shutdown recovery timer.

Syslog Messages

When MAC secure logging is configured and security violation occurs, syslog messages are displayed.

A sample syslog message is shown as follows. It indicates that the MAC is initially learnt on pseudowires (PW). When the same MAC is learnt on AC GigabitEthernet0_0_0_5.11, MAC secure configured under AC detects security violation. Then, the action configured under PW, which is shutdown in this case, is imposed on AC. Therefore, AC is brought down. If recovery timer is configured under AC, the AC is down until recovery timer expires. After the timer expires, AC is brought up.

```
%L2-L2FIB-5-SECURITY_MAC_SECURE_VIOLATION_AC : MAC secure in AC GigabitEthernet0_0_0_5.11
 detected violated packet that was previously learned in PW neighbor: 64.64.64.3,
ID: 2300001- source MAC: 0065.0600.0001, destination MAC: 0067.0600.0001; action: shutdown
```

MAC Address Move and Unicast Traffic Counters

MAC Address Move and Unicast Traffic counters are introduced on the VPLS bridge ports on the ASR9K platform. These counters essentially are L2VPN bridge port stats counters. MAC move and unicast traffic counters are introduced for troubleshooting. Cisco ASR 9000 High Density 100GE Ethernet Line Cards and Cisco ASR 9000 Enhanced Ethernet Line Cards support these counters.

For more information, on MAC Move and Unicast Traffic counters, use the **show l2vpn bridge-domain** command with the **detail** keyword on an AC Bridge, PW Bridge, PBB Edge, and VXLAN Bridge Ports.

Note If the bridge port traffic is forwarded, either completely or partially to ASR 9000 Ethernet line cards, MAC Address Move and Unicast Traffic counters may not be accurate.

LSP Ping over VPWS and VPLS

For Cisco IOS XR software, the existing support for the Label Switched Path (LSP) ping and traceroute verification mechanisms for point-to-point pseudowires (signaled using LDP FEC128) is extended to cover the pseudowires that are associated with the VFI (VPLS). Currently, the support for the LSP ping and traceroute for LDP signalled FEC128 pseudowires is limited to manually configured VPLS pseudowires. In addition, Cisco IOS XR software supports LSP ping for point-to-point single-segment pseudowires that are signalled using LDP FEC129 AII-type 2 applicable to VPWS or signalled using LDP FEC129 AII-type 1 applicable to VPLS. For information about Virtual Circuit Connection Verification (VCCV) support and the **ping mpls pseudowire** command, see the *MPLS Command Reference for Cisco ASR 9000 Series Routers*.

Split Horizon Groups

An IOS XR bridge domain aggregates attachment circuits (ACs) and pseudowires (PWs) in one of three groups called Split Horizon Groups. When applied to bridge domains, Split Horizon refers to the flooding and forwarding behavior between members of a Split Horizon group. The following table describes how frames received on one member of a split horizon group are treated and if the traffic is forwarded out to the other members of the same split horizon group.

Bridge Domain traffic is either unicast or multicast.

Flooding traffic consists of unknown unicast destination MAC address frames; frames sent to Ethernet multicast addresses (Spanning Tree BPDUs, etc.); Ethernet broadcast frames (MAC address FF-FF-FF-FF-FF-FF).

Known Unicast traffic consists of frames sent to bridge ports that were learned from that port using MAC learning.

Traffic flooding is performed for broadcast, multicast and unknown unicast destination address.

Table 5: Split Horizon Groups Supported in Cisco IOS-XR

Split Horizon Group	Who belongs to this Group?	Multicast within Group	Unicast within Group
0	Default—any member not covered by groups 1 or 2.	Yes	Yes
1	Any PW configured under VFI.	No	No
2	Any AC or PW configured with split-horizon keyword.	No	No

Important notes on Split Horizon Groups:

- All bridge ports or PWs that are members of a bridge domain must belong to one of the three groups.
- By default, all bridge ports or PWs are members of group 0.
- The VFI configuration submode under a bridge domain configuration indicates that members under this domain are included in group 1.
- A PW that is configured in group 0 is called an Access Pseudowire.
- The **split-horizon group** command is used to designate bridge ports or PWs as members of group 2.
- The ASR9000 only supports one VFI group.

Layer 2 Security

These topics describe the Layer 2 VPN extensions to support Layer 2 security:

Port Security

Use port security with dynamically learned and static MAC addresses to restrict a port's ingress traffic by limiting the MAC addresses that are allowed to send traffic into the port. When secure MAC addresses are assigned to a secure port, the port does not forward ingress traffic that has source addresses outside the group of defined addresses. If the number of secure MAC addresses is limited to one and assigned a single secure MAC address, the device attached to that port has the full bandwidth of the port.

These port security features are supported:

- Limits the MAC table size on a bridge or a port.
- Facilitates actions and notifications for a MAC address.
- Enables the MAC aging time and mode for a bridge or a port.
- Filters static MAC addresses on a bridge or a port.
- Marks ports as either secure or nonsecure.
- Enables or disables flooding on a bridge or a port.

After you have set the maximum number of secure MAC addresses on a port, you can configure port security to include the secure addresses in the address table in one of these ways:

- Statically configure all secure MAC addresses by using the **static-address** command.
- Allow the port to dynamically configure secure MAC addresses with the MAC addresses of connected devices.
- Statically configure a number of addresses and allow the rest to be dynamically configured.

Dynamic Host Configuration Protocol Snooping

Dynamic Host Configuration Protocol (DHCP) snooping is a security feature that acts like a firewall between untrusted hosts and trusted DHCP servers. The DHCP snooping feature performs these activities:

- Validates DHCP messages received from untrusted sources and filters out invalid messages.
- Rate-limits DHCP traffic from trusted and untrusted sources.
- Builds and maintains the binding database of DHCP snooping, which contains information about untrusted hosts with leased IP addresses.
- Utilizes the binding database of DHCP snooping to validate subsequent requests from untrusted hosts.

For additional information regarding DHCP, see the *Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Configuration Guide*.

G.8032 Ethernet Ring Protection

Ethernet Ring Protection (ERP) 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 pre-determined link or a failed link.

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 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

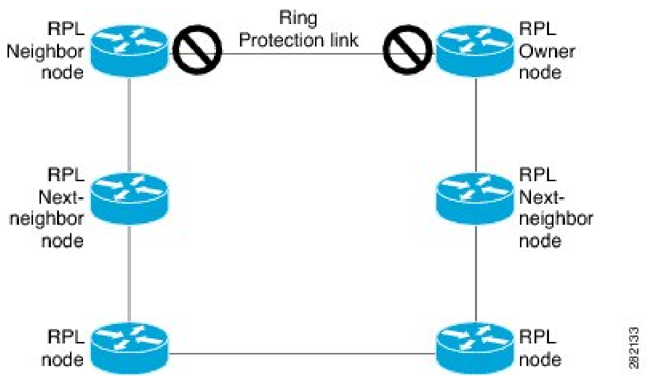
Loop avoidance in an Ethernet ring is achieved by ensuring that, at any time, traffic flows on all but one of the ring links which is the RPL. Multiple nodes are used to form a ring:

- RPL owner—It is 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 adjacent to the RPL. It is 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 adjacent to RPL owner node or RPL neighbor node. It is mainly used for FDB flush optimization on the ring. This node is also optional.

The following figure illustrates the G.8032 Ethernet ring.

Figure 26: 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 adjacent 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 adjacent 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 ERP 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 operator to forcefully block a particular ring-port.
  - Effective even if there is an existing SF condition
  - Multiple FS commands for ring supported
  - May be used to allow immediate maintenance operations
- Manual switch (MS)—Allows 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 packet can cross logical rings, and that is not desirable.

G.8032 ERP 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 is 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 ASR 9000 Series Router* module in the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide*.

Timers

G.8032 ERP 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, Wait-to-Restore (WTR) timer is used to verify that SF is not intermittent. The WTR timer can be configured by the operator, and the default time interval is 5 minutes. The time interval ranges from 1 to 12 minutes.
  - After FS/MS command, Wait-to-Block timer is used to verify that no background condition exists.

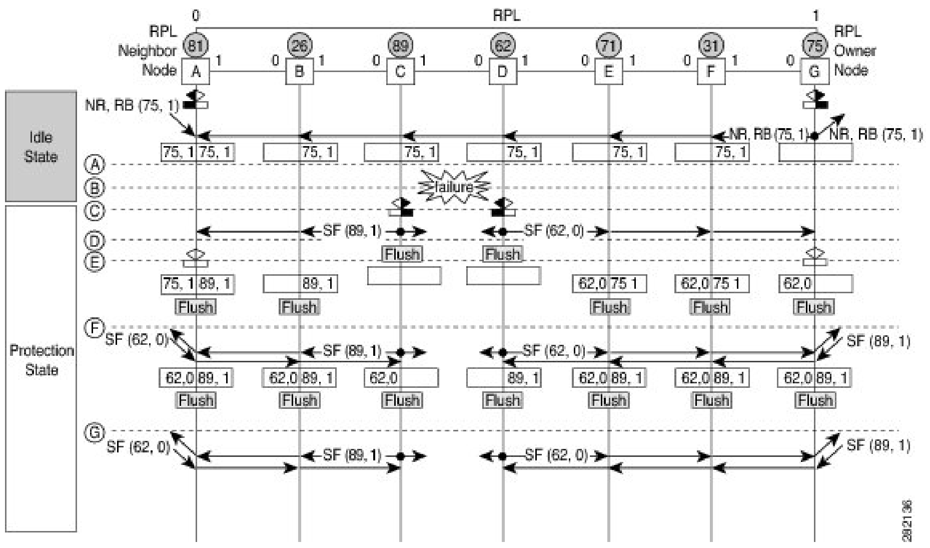
Note Wait-to-Block timer may be shorter than the Wait-to-Restore timer

- 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 from 10 to 2000 ms.
- Hold-off timers—used by 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 from 0 to 10 seconds.
  - Faults are reported to the ring protection mechanism, only if this timer expires.

Single Link Failure

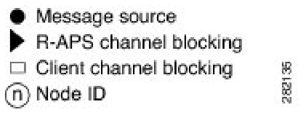
The following figure represents protection switching in case of a single link failure.

Figure 27: G.8032 Single Link Failure



The following figure represents an Ethernet ring composed of seven Ethernet ring nodes. The RPL is the ring link between Ethernet ring nodes A and G. In these scenarios, both ends of the RPL are blocked. Ethernet ring node G is the RPL owner node, and Ethernet ring node A is the RPL neighbor node.

These symbols are used:

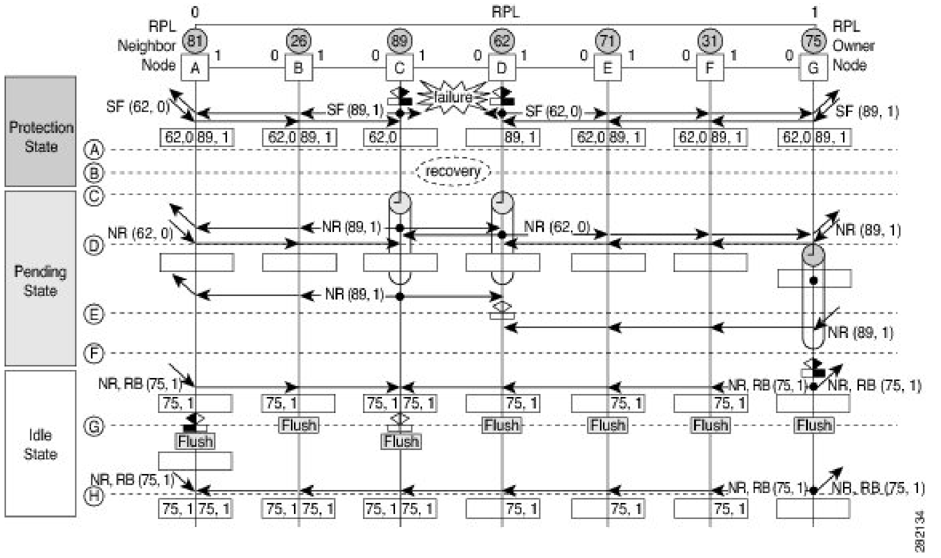


This sequence describes the steps in the single link failure, represented in Figure 8:

1. Link operates in the normal condition.
2. A failure occurs.
3. Ethernet ring nodes C and D detect a local Signal Failure condition and after the holdoff time interval, block the failed ring port and perform the FDB flush.
4. Ethernet ring nodes C and D start sending RAPS (SF) messages periodically along with the (Node ID, BPR) pair on both ring ports, while the SF condition persists.
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 it's 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.

The following figure represents reversion in case of a single link failure.

Figure 28: Single link failure Recovery (Revertive operation)



This sequence describes the steps in the single link failure recovery, as represented in Figure 9:

1. Link operates in the stable SF condition.
2. Recovery of link failure occurs.
3. Ethernet ring nodes C and D detect clearing of signal failure (SF) condition, start the guard timer and initiate periodical transmission of RAPS (NR) messages on both ring ports. (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 WTR timer expires, the RPL owner node blocks its end of the RPL, sends RAPS (NR, RB) message with the (Node ID, 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.

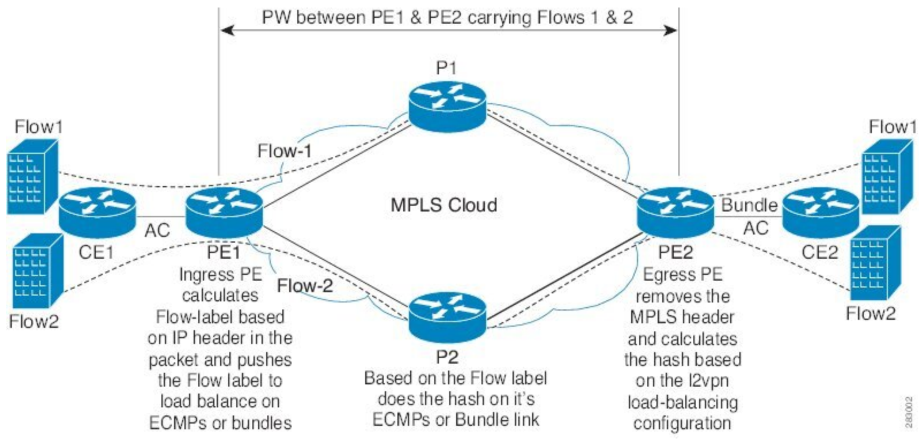
Flow Aware Transport Pseudowire (FAT PW)

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

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

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

Figure 29: FAT PW with two flows distributing over ECMPs and Bundle-Links


PW between PE1 & PE2 carrying Flows 1 & 2
P1
Flow1
Flow-1
MPLS Cloud
Flow1
Bundle
AC
AC
CE1
PE1
PE2
CE2
Ingress PE calculates Flow-label based on IP header in the packet and pushes the Flow label to load balance on ECMPs or bundles
Flow-2
Egress PE removes the MPLS header and calculates the hash based on the l2vpn load-balancing configuration
Flow2
Flow2
P2
Based on the Flow label does the hash on it's ECMPs or Bundle link
283002

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

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

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

Pseudowire Headend

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
PW Headend - GIL interface scale enhancement from 8 to 32	Release 7.3.1	This feature is now enhanced to support the maximum number of interfaces per generic interface list from 8 to 32 L3 physical interfaces.

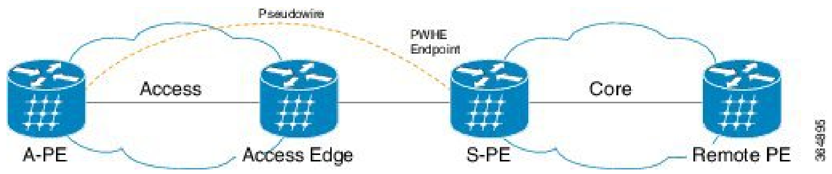
Pseudowires (PWs) enable payloads to be transparently carried across IP/MPLS packet-switched networks (PSNs). PWs are regarded as simple and manageable lightweight tunnels for returning customer traffic into core networks. Service providers are now extending PW connectivity into the access and aggregation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

Pseudowire Headend (PWHE) is a technology that allows termination of access pseudowires (PWs) into a Layer 3 (VRF or global) domain or into a Layer 2 domain. PWs provide an easy and scalable mechanism for tunneling customer traffic into a common IP/MPLS network infrastructure. PWHE allows customers to provision features such as QOS access lists (ACL), L3VPN on a per PWHE interface basis, on a service Provider Edge (PE) router.

Note Encapsulation default is not supported by PWHE.

Consider the following network topology as an example.

Figure 30: Pseudowire Network



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

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

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

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

Benefits of PWHE

Some of the benefits of implementing PWHE are:

- 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 UNI interface 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)

Restrictions

- PWHE over 2nd level BGP Labeled Unicast (LU) recursion (BGP RFC3107) supports only if there's a single BGP path to a destination.
- PHWE doesn't support BGP Prefix Independent Convergence (PIC) backup paths.
- PWHE can support only a single BGP Next-hop to pseudowire destination.
- PWHE supports up to one level of BGP recursion.
- PWHE with remote destinations(A-PE) reachable via BGP, need to configure a unique Generic Interface Lists (GILs) per A-PE through the same GIL members. For example, Using the same GIL1 for both PWHE#1 and PWHE#2 to reach BGP A-PE1 and BGP A-PE2 is not supported.
- PHWE cannot exceed more than eight Generic Interface Lists (GILs) through the same IGP peer, with the same or different GIL members, in the case of A-PE reachable through BGP.
- PHWE cannot exceed more than eight Generic Interface Lists (GILs) in the case of A-PE reachable through IGP.
- If you're running IGP to reach PWHE neighbor (A-PE), then we can apply the same GIL name on multiple PWHE.

Generic Interface List

A generic interface list contains is a list of physical or bundle interfaces that is used in a PW-HE connection.

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

A generic interface list 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 generic interface list and expects that the PWHE packets arrive on only line cards with generic interface list members on it. If packets arrive at the line card without generic interface list members on it, they are dropped.

Starting with Cisco IOS XR Software Release 7.3.1, the maximum number of interfaces per generic interface list is increased to 32 L3 physical interfaces. A maximum of eight L3 bundle interfaces is still supported. Furthermore, if there are more than eight interfaces under one of the generic interface lists, the supported number of generic interface list per IGP peer is reduced from eight to two.

Configure only those Layer 3 interfaces, either Bundle-Ether or physical, that are part of the generic interface list (GIL) through which PWHE connects access PE devices. PWHE does not support any Layer 2 interface or bundle member interfaces of the L3 interfaces list or L3VPN core-facing L3 interfaces.

However, the device still allows you to associate individual bundle member interfaces that are not part of the generic interface list (GIL). PWHE interfaces are up, and correspondingly all Xconnect configured on PWHE interfaces are up. But the traffic does not flow due to an undetected misconfiguration.

LFA over Pseudowire Headend

From Release 5.1.1, PW-HE is supported on loop free alternate (LFA) routes.

For LFA to be effective on a PW-HE interface, all the routing paths (protected and backup) must be included in the generic interface list of that PW-HE interface. If all the routing paths are not included in the generic interface list, then there may be loss of traffic even if LFA is enabled. This is because the LFA route could be the one that was not included in the generic interface list.

To configure LFA on PW-HE interface, do the following:

1. Configuring Generic Interface List
2. Configuring IP/LDP Fast Reroute

For more information about configuring the IP Fast Reroute Loop-free alternate, see Implementing IS-IS on Cisco IOS XR Software module of the *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide.*

PW-HE Multicast

Multicast support for Pseudowire Head-end (PW-HE) interfaces is available only on the enhanced ethernet cards.

For more information about PW-HE multicast feature, see the *Implementing Layer 3 Multicast Routing* chapter in the *Cisco ASR 9000 Series Aggregation Services Router Multicast Configuration Guide, Release 5.1.x*.

PW-HE over MPLS-TE Tunnels

The PW-HE over MPLS-TE Tunnels feature supports forwarding of pseudowire traffic (with pseudowire headend) over TE tunnels.

For more information about PW-HE over MPLS TE Tunnels, see the *Implementing MPLS Traffic Engineering* chapter in the *Cisco ASR 9000 Series Aggregation Services Router MPLS Configuration Guide, Release 5.1.x*.

PWHE Load Balancing Support using FAT Label

The PWHE Load Balancing Support using FAT Label feature enables flow hashing based load balancing across egress interfaces defined in the generic interface list. This feature also enables Flow Aware Transport (FAT) label that can be utilized for load balancing by downstream P-routers.

For load balancing among the members of the generic interface list, a five-tuple hash that includes source IP address, source port, destination IP address, destination port, and Router ID is utilized to distribute traffic among the members of the generic interface list.

You can insert the FAT label after successful negotiation with the remote router, or after static configuration. The downstream router uses it to perform load balancing. In Cisco IOS Software Release 6.6.1, only targeted-LDP signaling is supported to negotiate flow label.

Restrictions

- To implement the PWHE Load Balancing feature, use the Cisco ASR 9000 3rd generation line cards as your S-PE node ingress and egress line cards, because line card support is limited to only the 3rd generation.
- Flow labels are not supported when the incoming traffic is from an L2 transport PWHE interface (From Core to S-PE).
- Load balancing is not supported for MPLS-TE/TE preferred path.

- Enabling PWHE per-flow load balancing results in QoS behavior that is identical to multi-member bundles. A QoS policy instance is independently applied to each generic interface.

  Until Cisco IOS XR Software Release 6.6.1, only PW label hashing was supported, and absolute rate egress shaping and QoS could be configured. From Release 6.6.1, you can do one of the following:

  - Preserve the existing QoS accuracy without changing the load balancing mode.
  - Configure flow-based load balancing to take advantage of ECMP links at the detriment of absolute rate egress shaping and QoS.

- Configuring the **load-balancing flow src-dst-mac** command has the same effect as configuring **load-balancing flow src-dst-ip** command.
- This feature is not supported on PW-IW (VC type 11) interface.
- PWHE per-flow load-balancing is supported over L3 PW-Ethernet interfaces and sub-interfaces in L3, L3VPN and BNG deployment scenarios.
- PWHE per-flow load-balancing is not supported in EVPN, SR-TE and SR deployment scenarios.

Configure PWHE Load Balancing Support using FAT Label

This section describes how to configure PWHE load balancing using FAT label.

Configuration Example

Perfrom these tasks to configure PWHE load balancing using FAT label at both S-PE and A-PE.

S-PE Configuration

Perform these tasks at S-PE:

- Configure PWHE Ethernet interface and attach the generic interface list with a PWHE Ethernet interface.
- Configure global L2VPN load balancing (optional)
- Configure flow label
- Configure PWHE cross-connect

```
/* S-PE Configuration */

/* Configure PWHE Ethernet interface and attach the generic interface list with a PWHE
Ethernet interface */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# generic-interface-list gil1
RP/0/RSP0/CPU0:router(config-gen-if-list)# interface tenGigE 0/0/0/0/0
RP/0/RSP0/CPU0:router(config-gen-if-list)# interface tenGigE 0/1/0/1/0
RP/0/RSP0/CPU0:router(config-gen-if-list)# interface tenGigE 0/0/0/0/2
RP/0/RSP0/CPU0:router(config-gen-if-list)# exit
RP/0/RSP0/CPU0:router(config)# interface pw-ether 100
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 192.0.2.1/24
RP/0/RSP0/CPU0:router(config-if)# ipv6 address 2001:DB8::1/32
RP/0/RSP0/CPU0:router(config-if)# attach generic-interface-list gil1

/* Configure global L2VPN load balancing (Optional) */

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

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# load-balancing flow src-dst-ip

/* This enables flow-based load balancing across generic interface list members. */

/* Configure flow label */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class flow_lb_both
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)# load-balancing flow-label both

/* Configure PWHE cross-connect */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group pwhe_fat
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p xc_main_interface
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface PW-Ether100
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 209.165.200.225 pw-id 100
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class flow_lb_both
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit
```

A-PE Configuration

Perform these tasks at A-PE:

- Configure Ethernet L2 transport interface
- Configure flow label
- Configure PWHE cross-connect

```
/* A-PE Configuration */

/* Configure Ethernet L2 transport interface  */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface tenGigE 0/7/0/9.100 l2transport
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
RP/0/RSP0/CPU0:router(config-subif)# rewrite ingress tag pop 1 symmetric

/* Configure flow label */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class flow_lb_both
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)# load-balancing flow-label both

/* Configure PWHE cross-connect */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group pwhe_fat
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p xc_main_intf
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface tenGigE 0/7/0/9.100
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 209.165.201.1
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class flow_lb_both
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

This section shows PWHE load balancing using FAT label running configuration.

```
/* On S-PE */

/* Configure PWHE Ethernet interface and attach the generic interface list with a PWHE
Ethernet interface */

generic-interface-list gil1
 interface TenGigE0/0/0/0/0
 interface TenGigE0/0/0/0/2
 interface TenGigE0/1/0/1/0
!

interface PW-Ether100
 ipv4 address 192.0.2.1 255.255.255.0
 ipv6 address 2001:DB8::1/32
  attach generic-interface-list gil1
!

/* Configure global L2VPN load balancing */

l2vpn
 load-balancing flow src-dst-ip
!

/* Configure flow label */

l2vpn
 pw-class flow_lb_both
  encapsulation mpls
   load-balancing flow-label both
   !
  !

/* Configure PWHE cross-connect */

l2vpn
 xconnect group pwhe_fat
  p2p xc_main_interface
   interface PW-Ether100
   neighbor ipv4 209.165.200.225 pw-id 100
    pw-class flow_lb_both
   !
  !
!
```

```
/* On A-PE */

/* Configure Ethernet L2 transport interface */

interface TenGigE0/7/0/9.100 l2transport
 encapsulation dot1q 100
rewrite ingress tag pop 1 symmetric
!

/* Configure flow label */

l2vpn
 pw-class flow_lb_both
  encapsulation mpls
   load-balancing flow-label both
   !
```

```
/* Configure PWHE cross-connect */

l2vpn
 xconnect group pwhe_fat
  p2p xc_main_intf
   interface TenGigE0/7/0/9.100
   neighbor ipv4 209.165.201.1 pw-id 100
    pw-class flow_lb_both
   !
  !
```

Verification

The show output given in the following section display the details of the configuration of PWHE Ethernet interface, PWHE cross-connect, flow label, load balancing and the status of their configuration .

```
RP/0/RSP0/CPU0:router#show l2vpn xconnect pw-id 100 detail

Group pwhe_fat, XC xc_main_interface, state is up; Interworking none
  AC: PW-Ether100, state is up
    Type PW-Ether
    Interface-list: gil1
    Replicate status:
    Te0/0/0/0/0: success
    Te0/0/0/0/2: success
    Te0/1/0/1/0: success
    MTU 1500; interworking none
    Internal label: 232235
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
  PW: neighbor 209.165.200.225, PW ID 100, state is up ( established )
    PW class flow_lb_both, XC ID 0xa0000013
    Encapsulation MPLS, protocol LDP
    Source address 209.165.201.1
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    LSP : Up
    Load Balance Hashing: src-dst-ip
    Flow Label flags configured (Tx=1,Rx=1), negotiated (Tx=1,Rx=1)

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        28130                          24114
      Group ID     0x40406e0                      0x12000300
      Interface    PW-Ether100                    TenGigE0/7/0/9.100
      MTU          1500                           1500
      Control word disabled                       disabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x6                            0x6
                   (RP/0/RSP0/CPU0:router alert label)           (RP/0/RSP0/CPU0: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: 2684354579
Create time: 25/09/2018 15:02:44 (1d04h ago)
Last time status changed: 26/09/2018 18:07:25 (01:41:29 ago)
Last time PW went down: 26/09/2018 17:45:25 (02:03:28 ago)
Statistics:
  packets: received 0, sent 0
  bytes: received 0, sent 0
```

Related Topics

- Flow Aware Transport Pseudowire (FAT PW) , on page 237
- Pseudowire Headend, on page 238

Associated Commands

- load-balancing flow
- xconnect group
- show l2vpn xconnect

MVPN Profile 26 for Pseudowire Headend

The MVPN Profile 26 for Pseudowire Headend feature adds support for multicast VPN (MVPN) profile 26 on pseudowire headend (PWHE) interface to provide multicast services. Profile 26 is a next-generation MVPN profile that carries multicast traffic encapsulated in MPLS packets. The traffic is forwarded with MPLS labels using Point-to-Multipoint (P2MP) MPLS Traffic Engineering (TE) as the core signaling protocol. BGP is used as the overlay signaling protocol for partitioned multicast distribution trees (MDTs). Profile 26 represents partitioned-P2MP-TE with BGP C-multicast routing.

Some of the characteristics of this profile include:

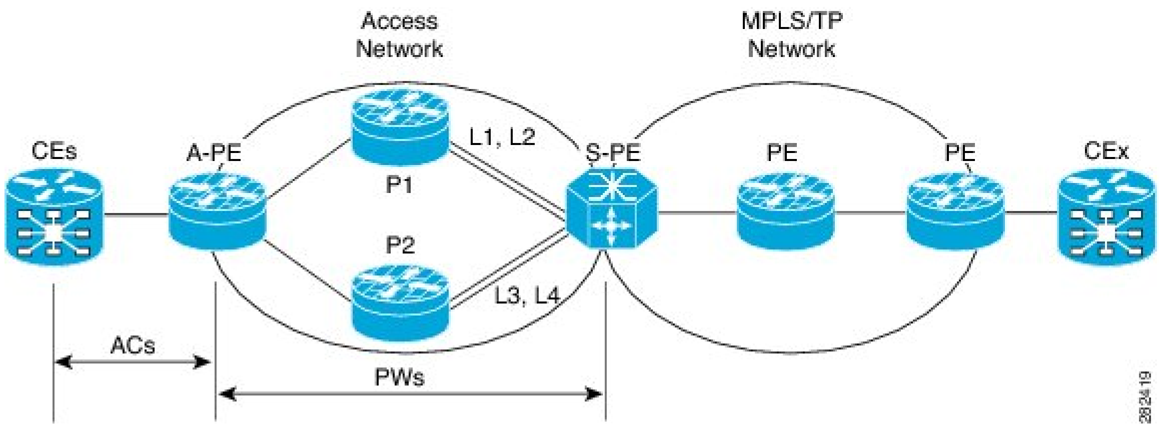
- Dynamic P2MP-TE tunnels with BGP C-multicast routing
- All Upstream Multicast Hop (UMH) options are supported
- Default and data MDT are supported
- Customer traffic can be SM or source-specific multicast (SSM)
- RIB-Extranet, RPL-Extranet, Hub and Spoke are supported
- Inter-AS option A and C are supported

Consider a topology where PWHE interfaces reside on the S-PE facing the access interface and is configured with the customer VRF. Access Pseudowires (PWs) from the A-PE of the Layer 2 domain terminate on this PWHE on the S-PE in to Layer 3 MVPN Profile 26 MPLS network. Psesudowire connectivity is established between A-PE and S-PE through cross-connect. Routing is established end-to-end through IGP protocols. The nodes are connected through MPLS core.

P2MP TE auto-tunnels are established dynamically when S-PE receives PIM Join message from the participating PE receivers.

Multicast traffic from the access side towards the PWHE interface on the S-PE travels across the MPLS core as encapsulated MPLS packets through P2MP TE auto-tunnels to the tail-end PEs. Similary, for multicast traffic with the sources from the L3 domain, tunnels are established with S-PE as the tail-end PE. The traffic is decapsulated in to the L2 domain to the access PE receivers.

Figure 31: MVPN Profile 26 for Pseudowire Headend



Partitioned MDT with BGP C-multicast Signaling

Partitioned MDT is an MDT between a subset of PE routers and an ingress PE router. Each egress PE router advertises itself by sending a Type 3 route. This BGP Type 3 route has wildcards, which indicates that it applies for any source and any group. The Type 3 route has the PTA indicating the core tree protocol to be used and the kind of tree. This type 3 route only indicates the possibility to become part of any partitioned MDT. No partitioned MDT is set up. When a PE receives a PIM Join for C-(S,G) , the PE performs a reverse path forwarding (RPF) lookup to find the BGP next-hop address. This PE joins the tree as indicated in the Type 3 route towards the root which is the BGP next-hop. The S-PE forwards the C-(S,G) traffic onto the tree.

With P2MP TE as the core tree protocol, the P2MP TE label switched-path (LSP) must be set up by the head-end link-state routing (LSR). In this case it is the ingress PE router. Egress PE routers nofitify the ingress PE that they would be tail end routers of a new P2MP TE, which is the partitioned MDT. Egress PE routers send a BGP AD Route Type 4 targetted at ingress PE. Ingress PE learns that there are two tail end routers for the new P2MP TE LSP and initiates the signaling of the P2MP TE LSP with RSVP. Ingress sends out path messages towards tail end routers. Egress PEs send back received messgaes towards the ingress PE. When this signaling is successful, the partitioned MDT comprising of a P2MP TE LSP, is set up.

To build up the partitioned MDT with P2MP TE, BGP-AD is used utilizing the address family IPv4 MVPN. This way each PE can become the MPLS TE head-end router of a P2MP MPLS TE tree or tunnel. Also, learn which routers are the tail-end routers, and proceed with the RSVP-TE signaling towards each of the tail-end routers. These tunnels are P2MP auto-tunnels that are provisioned automatically.

Configure MVPN Profile 26 for Pseudowire Headend

Perform the following tasks to configure MVPN Profile 26 for Pseudowire Headend feature:

S-PE Configuration

- Configure Profile 26
- Configure generic interface list
- Configure PWHE Ethernet and interworking interfaces, attach the generic interface list with a PWHE Ethernet and interworking interfaces

- Configure PW class for Ethernet and interworking interfaces
- Configure cross-connect using PWHE Ether and PWIW interfaces

```
/* S-PE Configuration */

/* Configure Profile 26 */
Router#configure
Router(config)#router pim
Router(config-pim)#vrf one
Router(config-pim-vrf1)#address-family ipv4
Router(config-pim-vrf1-ipv4)#rpf topology route-policy rpf-vrf-one
Router(config-pim-vrf1-ipv4)#mdt c-multicast-routing bgp`
Router(config-pim-vrf1-ipv4-md-cmcast)#exit
Router(config-pim-vrf1)#address-family ipv6
Router(config-pim-vrf1-ipv6)#rpf topology route-policy rpf-vrf-one
Router(config-pim-vrf1-ipv6)#mdt c-multicast-routing bgp
Router(config-pim-vrf1-ipv6-md-cmcast)#commit

Router(config)#interface GigabitEthernet
Router(config-if)#route-policy rpf-vrf-one
Router(config-rpl)#set core-tree p2mp-te-partitioned
Router(config-rpl)#end-policy


Router#configure
Router(config)#multicast-routing
Router(config-mcast)#vrf one
Router(config-mcast-vrf1)#address-family ipv4
Router(config-mcast-vrf1-ipv4)#rate-per-route
Router(config-mcast-vrf1-ipv4)#interface all enable`
Router(config-mcast-vrf1-ipv4)#accounting per-prefix
Router(config-mcast-vrf1-ipv4)#bgp auto-discovery p2mp-te
Router(config-mcast-vrf1-ipv4-bgp-ad)#mdt source Loopback0
Router(config-mcast-vrf1-ipv4-bgp-ad)#mdt partitioned p2mp-te 100
Router(config-mcast-vrf1-ipv4-bgp-ad)#exit

Router(config-mcast-vrf1)#address-family ipv6
Router(config-mcast-vrf1-ipv6)#rate-per-route
Router(config-mcast-vrf1-ipv6)#interface all enable`
Router(config-mcast-vrf1-ipv6)#accounting per-prefix
Router(config-mcast-vrf1-ipv6)#bgp auto-discovery p2mp-te
Router(config-mcast-vrf1-ipv6-bgp-ad)#mdt source Loopback0
Router(config-mcast-vrf1-ipv6-bgp-ad)#mdt partitioned p2mp-te 100
Router(config-mcast-vrf1-ipv6-bgp-ad)#commit

/* Configure generic interface list for PWHE Ethernet interface */

Router# configure
Router(config)# generic-interface-list pwhe-list-APE2-1
Router(config-gen-if-list)# interface Bundle-Ether200
Router(config-gen-if-list)# interface TenGigE0/0/0/0/4
Router(config-gen-if-list)# interface TenGigE0/0/0/0/5


/* Configure PWHE Ethernet interface and attach the generic interface list with a PWHE
Ethernet interface */

Router# configure
Router(config)# interface pw-ether 5001
Router(config-if)# ipv4 address 103.7.7.1 255.255.255.252
Router(config-if)# ipv6 address 103:107:1::1/126
Router(config-if)# attach generic-interface-list pwhe-list-APE2-1


/* Configure generic interface list for PW interworking interface */

Router# configure
Router(config)# generic-interface-list pwhe-list-APE-2
Router(config-gen-if-list)# interface Bundle-Ether201
Router(config-gen-if-list)# interface TenGigE0/0/0/0/6
Router(config-gen-if-list)# interface TenGigE0/0/0/0/7


/* Configure interworking interface and attach the generic interface list with an interworking
 interface */

Router# configure
Router(config)#  interface pw-iw 4001
Router(config-if)# ipv4 address 103.107.47.225 255.255.255.252
Router(config-if)# attach generic-interface-list pwhe-list-APE-2


/* Configure PW class for Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  pw-class APE2-PE1-PORT
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet


/* Configure cross-connect for PW Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE2-PE1-PORT
Router(config-l2vpn-xc)# p2p APE2-PE1-5001
Router(config-l2vpn-xc-p2p)# interface PW-Ether5001
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.8.1 pw-id 5001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE2-PE1-PORT


/* Configure PW class for interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word


/* Configure cross-connect for PW interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE-InetRI-PWIW-4001
Router(config-l2vpn-xc)# p2p APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p)# interface PW-IW4001
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.9.1 pw-id 4001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p-pw)# interworking ipv4
```

A-PE Configuration

- Configure PWHE Ethernet and interworking interfaces, attach the generic interface list with a PWHE Ethernet and interworking interfaces
- Configure PW class for Ethernet and interworking interfaces
- Configure cross-connect using PWHE Ether and PWIW interfaces

```
/* A-PE Configuration */

/* Configure PWHE  Ethernet interface */

Router# configure
Router(config)# interface TenGigE0/0/0/0 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config-subif)# rewrite ingress tag pop 1 symmetric


/* Configure interworking interface */

Router# configure
Router(config)# interface Serial0/5/0/0/8 l2transport



/* Configure PW Class for Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class APE2-PE1-PORT
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet


/* Configure Cross-connect for Ethernet interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE2-PE1-PORT
Router(config-l2vpn-xc)# p2p APE2-PE1-5001
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/0
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.1.1 pw-id 5001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE2-PE1-PORT


/* Configure PW Class for interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# control-word



/* Configure Cross-connect for interworking interface */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group APE-InetRI-PWIW-4001
Router(config-l2vpn-xc)# p2p APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p)# interface Serial0/5/0/0/8
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 100.1.1.1 pw-id 4001
Router(config-l2vpn-xc-p2p-pw)# pw-class APE-InetRI-PWIW-4001
Router(config-l2vpn-xc-p2p-pw)# interworking ipv4
```

Running Configuration

This section shows MVPN Profile 26 for Pseudowire Headend running configuration.

```
/* On S-PE */

/* Profile 26 */
vrf one
  address-family ipv4
   rpf topology route-policy rpf-vrf-one
   mdt c-multicast-routing bgp
   !
  address-family ipv6
   rpf topology route-policy rpf-vrf-one
   mdt c-multicast-routing bgp
   !

interface GigabitEthernet
 route-policy rpf-vrf-one
  set core-tree p2mp-te-partitioned
end-policy

multicast-routing
 vrf one
  address-family ipv4
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery p2mp-te
    mdt source Loopback0
    mdt partitioned p2mp-te 100
    !
   !
  address-family ipv6
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery p2mp-te
    mdt source Loopback0
    mdt partitioned p2mp-te 100

/* Ethernet interface */

configure
 generic-interface-list pwhe-list-APE2-1
  interface Bundle-Ether200
  interface TenGigE0/0/0/0/4
  interface TenGigE0/0/0/0/5
  !
!

configure
 interface PW-Ether5001
  ipv4 address 103.107.1.1 255.255.255.252
  ipv6 address 103:107:1::1/126
  attach generic-interface-list pwhe-list-APE2-1

!
```

```
l2vpn
 pw-class APE2-PE1-PORT
  encapsulation mpls
   control-word
   transport-mode ethernet
  !

!

l2vpn
 xconnect group APE2-PE1-PORT
  p2p APE2-PE1-5001
   interface PW-Ether5001
   neighbor ipv4 100.1.8.1 pw-id 5001
    pw-class APE2-PE1-PORT



/* Interworking interface */

configure
 generic-interface-list pwhe-list-APE-2
  interface Bundle-Ether200
  interface TenGigE0/0/0/0/6
  interface TenGigE0/0/0/0/7
  !
!


configure
 interface PW-IW4001
  ipv4 address 103.107.47.225 255.255.255.252
  attach generic-interface-list pwhe-APE-2

!

l2vpn
 pw-class APE-InetRI-PWIW-4001
  encapsulation mpls
   control-word
   !
!

l2vpn
 xconnect group APE-InetRI-PWIW-4001
  p2p APE-InetRI-PWIW-4001
   interface PW-IW4001
   neighbor ipv4 100.1.9.1 pw-id 4001
    pw-class APE-InetRI-PWIW-4001
    !
    interworking ipv4
    !
  !

 !
```

```
/* On A-PE */

/* Ethernet interface */

configure
```

```
 interface TenGigE0/0/0/0 l2transport
  encapsulation dot1q 1001
  rewrite ingress tag pop 1 symmetric
!

l2vpn
 pw-class APE2-PE1-PORT
  encapsulation mpls
   control-word
   transport-mode ethernet
  !
!

l2vpn
 xconnect group APE2-PE1-PORT
  p2p APE2-PE1-5001
   interface TenGigE0/0/0/0
   neighbor ipv4 100.1.1.1 pw-id 5001
    pw-class APE2-PE1-PORT
   !
!


/* Interworking interface */

configure
 interface Serial0/5/0/0/8 l2transport
 !
!

l2vpn
 pw-class APE-InetRI-PWIW-4001
  encapsulation mpls
  control-word
  !
!

l2vpn
 xconnect group APE-InetRI-PWIW-4001
  p2p APE-InetRI-PWIW-4001
   interface Serial0/5/0/0/8
   neighbor ipv4 100.1.1.1 pw-id 4001
   pw-class APE-InetRI-PWIW-4001
   !
   interworking ipv4
   !
  !
```

Verification

The show outputs given in the following section display the details of the configuration of PW Ethernet interface and cross-connect, and the status of their configuration on S-PE and A-PE.

```
/* S-PE Configuration */

Router-S-PE#  show l2vpn xconnect interface pw-ether 5001 detail
Group APE2-PE1-PORT, XC APE2-PE1-5001, state is up; Interworking none
  AC: PW-Ether5001, state is up
    Type PW-Ether
    Interface-list: pwhe-list-APE2-1
    Replicate status:
```

```
    BE616: success
    BE617: success
    MTU 8986; interworking none
    Internal label: 25002
    Statistics:
      packets: received 96860, sent 101636
      bytes: received 7285334, sent 8703696
  PW: neighbor 100.1.8.1, PW ID 5001, state is up ( established )
    PW class APE2-PE1-PORT, XC ID 0xfffe07d0
    Encapsulation MPLS, protocol LDP
    Source address 100.1.1.1
    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        29012                          24001
      Group ID     0x4607d3c                      0x40000c0
      Interface    PW-Ether5001                   TenGigE0/0/0/0
      MTU          8986                           8986
      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: 4294838224
    Create time: 26/09/2017 11:08:57 (18:23:34 ago)
    Last time status changed: 26/09/2017 11:28:59 (18:03:32 ago)
    Statistics:
      packets: received 96860, sent 101636
      bytes: received 7285334, sent 8703696
```

```
/* A-PE configuration details */

Router-A-PE# show l2vpn xconnect interface te0/0/0/0 detail
Group APE2-PE1-PORT, XC APE2-PE1-5001, state is up; Interworking none
  AC: TenGigE0/0/0/0, state is up
    Type Ethernet
    MTU 8986; XC ID 0x1084455; interworking none
    Statistics:
      packets: received 399484457, sent 1073874787256
      bytes: received 42812068782, sent 81549786107821
  PW: neighbor 100.1.1.1, PW ID 5001, state is up ( established )
    PW class APE2-PE1-PORT, XC ID 0xc0000fa1
    Encapsulation MPLS, protocol LDP
    Source address 100.1.8.1
    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        24001                          29012
      Group ID     0x40000c0                      0x4607d3c
      Interface    TenGigE0/0/0/0                 PW-Ether5001
      MTU          8986                           8986
      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: 3221229473
    Create time: 04/09/2017 17:48:35 (3w1d ago)
    Last time status changed: 26/09/2017 23:37:11 (18:01:16 ago)
    Last time PW went down: 26/09/2017 23:21:53 (18:16:34 ago)
    Statistics:
      packets: received 1073874787256, sent 399484457
      bytes: received 81549786107821, sent 42812068782
```

The show outputs given in the following section display the details of the configuration of PW interworking interface and cross-connect, and the status of their configuration on S-PE and A-PE.

```
/* S-PE Configuration */

Router-S-PE#  show l2vpn xconnect interface pw-iw 4001 detail
Group APE-InetRI-PWIW-4001, XC APE-InetRI-PWIW-4001, state is up; Interworking IPv4
  AC: PW-IW4001, state is up
    Type PW-IW
    Interface-list: pwhe-APE-2
    Replicate status:
    BE616: success
    MTU 4470; interworking IPv4
    Internal label: 35423
    Statistics:
      packets: received 185986, sent 185985
      bytes: received 134084287, sent 134654943
  PW: neighbor 100.1.9.1, PW ID 4001, state is up ( established )
    PW class APE-InetRI-PWIW-4001, XC ID 0xfffe1058
    Encapsulation MPLS, protocol LDP
    Source address 100.1.1.1
    PW type IP, control word enabled, interworking IPv4
    PW backup disable delay 0 sec
    Sequencing not set

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        152284                         24018
      Group ID     0x84003ed4                     0xe004040
      Interface    PW-IW4001                      Serial0/5/0/0/8
      MTU          4470                           4470
      Control word enabled                        enabled
      PW type      IP                             IP
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x7                            0x7
                   (control word)                 (control word)
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ -----------------------------
    Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    MIB cpwVcIndex: 4294840408
    Create time: 04/10/2017 09:55:04 (00:21:36 ago)
    Last time status changed: 04/10/2017 10:02:45 (00:13:56 ago)
    Statistics:
      packets: received 185986, sent 185985
      bytes: received 134084287, sent 134654943
```

```
/* A-PE configuration details */

Router-A-PE# show interface pw-iw 4001
PW-IW4001 is up, line protocol is up
  Interface state transitions: 1
  Hardware is PWHE VC11 IP Interworking Interface
  Internet address is 103.107.47.225/30
  MTU 4470 bytes, BW 10000 Kbit (Max: 10000 Kbit)
     reliability 255/255, txload 7/255, rxload 7/255
  Encapsulation PW-IW,  loopback not set,
  Last link flapped 00:14:44
    L2Overhead: 0
    Generic-Interface-List: pwhe-APE-2
  Last input 00:11:04, output 00:11:04
  Last clearing of "show interface" counters never
  5 minute input rate 277000 bits/sec, 48 packets/sec
  5 minute output rate 293000 bits/sec, 51 packets/sec
     185986 packets input, 134084287 bytes, 0 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     185985 packets output, 134654943 bytes, 0 total output drops
     Output 0 broadcast packets, 0 multicast packets
```

The show output given in the following section display the details of the configuration of PWHE subinterface.

```
Router# show interface pw-ether 5001.1001
PW-Ether5001.1001 is up, line protocol is up
  Interface state transitions: 1
  Hardware is VLAN sub-interface(s), address is ac19.7200.0001
  Internet address is 105.1.1.1/30
  MTU 9004 bytes, BW 10000 Kbit (Max: 10000 Kbit)
     reliability 255/255, txload 0/255, rxload 0/255
  Encapsulation 802.1Q Virtual LAN, VLAN Id 1001,  loopback not set,
  Last link flapped 00:14:56
  ARP type ARPA, ARP timeout 04:00:00
  Last input 00:00:00, output 00:00:00
  Last clearing of "show interface" counters never
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 1000 bits/sec, 0 packets/sec
     196 packets input, 15554 bytes, 1282 total input drops
     0 drops for unrecognized upper-level protocol
     Received 0 broadcast packets, 0 multicast packets
     1416 packets output, 923195 bytes, 3 total output drops
```

```
Output 0 broadcast packets, 15 multicast packets
```

DHCP on PWHE Interfaces

The DHCP on PWHE interfaces feature enables you to configure the DHCP functionality, such as DHCP server, relay, or proxy on the PWHE interface.

This feature allows you to offer LAN IP gateway services, such as, DHCP and DNS to your customers connected through PWHE interface. This feature removes the need for an extra layer 3 router at your branch site. Instead, use S-PE to centralize the DHCP-server functionality to lower the cost and simplify on-premise solution. This feature minimizes the equipment needed on-premise at the end-customer premises. Centralizing DHCP on the S-PE allows you to offer faster time-to-market because you do not have to ship layer 3 devices to your customer site. This feature significantly simplifies the connectivity of enterprise branch sites requiring LAN DHCP services from their service provider.

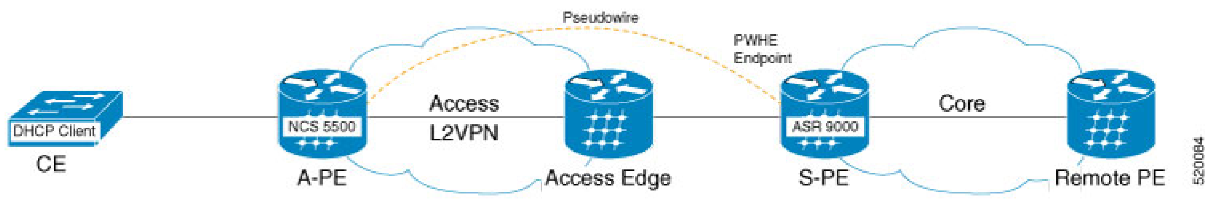
This feature is supported only on 64-bit IOS XR operating system.

Consider a topology where an NCS 5500 router is facing the access side, and an ASR 9000 router is facing the core side. Create a PWHE tunnel between these routers. The PWHE tunnel originates at L2 on the NCS 5500 router and terminates at L3 on the ASR 9000 router. Configure EVPN-VPWS cross-connect between the NCS 5500 router and the ASR 9000 router.

You can configure the ASR 9000 router as a DHCP server, relay, or proxy on the PWHE main interface or sub-interface.

The AC carries the traffic in its native form; that is, Ethernet frames with or without VLAN tagging, depending on whether you are creating a VLAN-based pseudowire or an Ethernet-based pseudowire.

Figure 32: DHCP Functionality for PWHE Interfaces



If you configure the ASR 9000 router as a DHCP server on the PWHE interface, it processes the request received from the DHCP client. The DHCP server accepts address assignment requests and renewals, and assigns the IP addresses from predefined groups of addresses contained within Distributed Address Pools (DAPS). You can also configure the DHCP server to provide additional information to the requesting client, such as subnet mask, domain-name, the IP address of the DNS server, the default router, and other configuration parameters.

If you configure the ASR 9000 router as a DHCP relay or proxy on the PWHE interface, it forwards the request received from the DHCP client to the remote server. The remote server processes the request and provides information back to the DHCP relay or proxy.

This feature supports both DHCP IPv4 and IPv6 addresses.

Configure DHCP on Pseudowire Headend

To configure DHCP on Pseudowire Headend feature, perform the following tasks:

- Configure PWHE on S-PE and A-PE

- Configure DHCP on S-PE

Configure PWHE on both S-PE and A-PE

This section describes how you can configure pseudowire headend on both S-PE and A-PE.

S-PE Configuration

```
/* S-PE Configuration */

Router# configure
Router(config)# generic-interface-list pwhe-list-pwhe2
Router(config-gen-if-list)# interface pw-ether 2
Router(config-gen-if-list)# exit
Router# configure
Router(config)# interface pw-ether 2
Router(config-if)# ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)# attach generic-interface-list pwhe-list-pwhe2
Router(config-if)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p vpws_1
Router(config-l2vpn-xc-p2p)# PW-Ether2
Router(config-l2vpn-xc-p2p-pw)# neighbor evpn evi 25 target 200 source 100
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config)# generic-interface-list pwhe2
Router(config-gen-if-list)#  interface TenGigE0/1/0/1
```

A-PE Configuration

```
/* A-PE Configuration */

Router# configure
Router(config)# interface TenGigE0/0/0/0 l2transport
Router(config-subif)# encapsulation dot1q 1001
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p vpws_1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/21
Router(config-l2vpn-xc-p2p-pw)# neighbor evpn evi 25 target 100 source 200
```

Running Configuration

This section shows pseudowire headend running configuration.

```
/* On S-PE */

configure
 generic-interface-list pwhe-list-pwhe2
  interface pw-ether 2

  !

 interface pw-ether 2
  ipv4 address 10.0.0.1 255.0.0.0
  attach generic-interface-list pwhe-list-pwhe2

!
```

```
l2vpn
 xconnect group evpn_vpws
  p2p vpws_1
   interface PW-Ether2
   neighbor evpn evi 25 target 200 source 100
   !
generic-interface-list pwhe2
 interface TenGigE0/1/0/1
```

```
/* On A-PE */

configure
 interface TenGigE0/0/0/0 l2transport
  encapsulation dot1q 1001
  rewrite ingress tag pop 1 symmetric
!
!
l2vpn
 xconnect group evpn_vpws
  p2p vpws_1
   interface TenGigE0/0/0/21
   neighbor evpn evi 25 target 100 source 200
   !
!
```

Verification

The show command outputs given in this section display the details of the configuration of PW Ethernet interface and cross-connect, and the status of their configuration.

```
Router#  show l2vpn xconnect

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
------------------------   ----------------------------    ----------------------------
evpn_vpws  vpws_1     UP   PE2                    UP       EVPN 25,172.16.0.1   UP
```

Configure DHCP on S-PE

You can configure the ASR 9000 series router as DHCP server, relay, or proxy on the PWHE main interface or sub-interface.

Configure DHCP Server

Perform these tasks to configure DHCP server on PWHE interface.

```
Router# configure
Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile svr1 server
Router(config-dhcpv4-server-profile)# lease lease 0 1 0
Router(config-dhcpv4-server-profile)# pool pool_1
Router(config-dhcpv4-server-profile)# exit
Router(config-dhcpv4)# profile svr2 server
Router(config-dhcpv4-server-profile)# lease lease 0 1 0
Router(config-dhcpv4-server-profile)# pool pool_2
Router(config-dhcpv4-server-profile)# exit
Router(config-dhcpv4)#interface PW-Ether2.101 server profile SVR1
Router(config-dhcpv4)#interface PW-Ether2.102 server profile SVR2
Router(config-dhcpv4)#exit
Router(config)# pool vrf default ipv4 pool_1
Router(config-pool-ipv4)# address-range 10.0.0.1 10.255.255.254
Router(config-pool-ipv4)# exit
Router(config)# pool vrf default ipv4 pool_2
Router(config-pool-ipv4)# address-range 192.168.0.1 192.168.255.254
Router(config-pool-ipv4)# exit
```

Running Configuration

This section shows DHCP server running configuration.

```
configure
 dhcp ipv4
  profile SVR1 server
  lease 0 1 0
  pool pool_1
 !
  profile SVR2 server
  lease 0 1 0
  pool pool_2
 !

 !
 interface PW-Ether2.101 server profile SVR1
 interface PW-Ether2.102 server profile SVR2
```

Verification

Verify the DHCP server configuration.

```
Router#show dhcp ipv4 server binding
Wed Nov 20 13:09:11.579 IST
                                          Lease

 MAC Address      IP Address      State    Remaining       Interface          VRF
Sublabel
--------------  --------------  ---------  ---------  -------------------  ---------
----------
0010.9400.c02a  10.0.0.22        BOUND      3099       PE2.102              default    0x0

0010.9400.c02b  10.0.0.23        BOUND      3099       PE2.102              default    0x0

0010.9400.c02c  10.0.0.24        BOUND      3099       PE2.102              default    0x0

0010.9400.c02d  10.0.0.25        BOUND      3099       PE2.102              default    0x0

0010.9400.c02e  10.0.0.26        BOUND      3099       PE2.102              default    0x0

0010.9400.c02f  10.0.0.27        BOUND      3099       PE2.102              default    0x0

0010.9400.c030  10.0.0.28        BOUND      3099       PE2.102              default    0x0
```

```
0010.9400.c031  10.0.0.29        BOUND      3099       PE2.102              default    0x0

0010.9400.c032  10.0.0.30        BOUND      3099       PE2.102              default    0x0

0010.9400.c033  10.0.0.11        BOUND      3099       PE2.102              default    0x0

0010.9400.c020  10.0.0.45        BOUND      3201       PE2.101              default    0x0

0010.9400.c021  10.0.0.46        BOUND      3201       PE2.101              default    0x0

0010.9400.c022  10.0.0.47        BOUND      3201       PE2.101              default    0x0

0010.9400.c023  10.0.0.48        BOUND      3201       PE2.101              default    0x0

0010.9400.c024  10.0.0.49        BOUND      3201       PE2.101              default    0x0

0010.9400.c025  10.0.0.50        BOUND      3201       PE2.101              default    0x0

0010.9400.c026  10.0.0.51        BOUND      3201       PE2.101              default    0x0

0010.9400.c027  10.0.0.52        BOUND      3201       PE2.101              default    0x0

0010.9400.c028  10.0.0.53        BOUND      3201       PE2.101              default    0x0

0010.9400.c029  10.0.0.54        BOUND      3201       PE2.101              default    0x0
```

Configure DHCP Proxy

Perform these tasks to configure DHCP proxy on PWHE interface.

```
Router# configure
Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile prox1 proxy
Router(config-dhcpv4-proxy-profile)# helper-address vrf default 10.0.0.1 giaddr 0.0.0.0
Router(config-dhcpv4-proxy-profile)# relay information check
Router(config-dhcpv4-proxy-profile)# relay information option
Router(config-dhcpv4-proxy-profile)# relay information option allow-untrusted
Router(config-dhcpv4-proxy-profile)# exit
Router(config-dhcpv4)#interface PW-Ether2.201 proxy profile prox
Router(config-dhcpv4)#interface PW-Ether2.202 proxy profile prox
Router(config-dhcpv4)#commit
```

Running Configuration

This section shows DHCP proxy running configuration.

```
configure
 dhcp ipv4
  profile prox1 proxy
   helper-address vrf default 10.0.0.1 giaddr 0.0.0.0
   relay information check
   relay information option
   relay information option allow-untrusted
 !
 !

 interface PW-Ether2.101 proxy profile prox
 interface PW-Ether2.102 proxy profile prox
```

Verification

Verify DHCP proxy configuration.

```
Router#show dhcp ipv4 proxy binding summary
Wed Jun  5 08:07:22.537 IST

Total number of clients: 3724

     STATE                |     COUNT     |
------------------------------------
  INIT                    |            0  |
  INIT_DPM_WAITING        |            0  |
  SELECTING               |            0  |
  OFFER_SENT              |            0  |
  REQUESTING              |            0  |
  REQUEST_INIT_DPM_WAITING|            0  |
  ACK_DPM_WAITING         |            0  |
  BOUND                   |         3724  |
  RENEWING                |            0  |
  INFORMING               |            0  |
  REAUTHORIZE             |            0  |
  DISCONNECT_DPM_WAIT     |            0  |
  ADDR_CHANGE_DPM_WAIT    |            0  |
  DELETING                |            0  |
  DISCONNECTED            |            0  |
  RESTARTING              |            0  |
```

Configure DHCP Relay

Perform these tasks to configure DHCP relay on PWHE interface.

```
Router# configure
Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile relay1 relay
Router(config-dhcpv4-proxy-profile)# helper-address vrf default 10.0.0.1 giaddr 0.0.0.0
Router(config-dhcpv4-proxy-profile)# relay information option vpn
Router(config-dhcpv4-proxy-profile)# relay information check
Router(config-dhcpv4-proxy-profile)# relay information option
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
Router(config-dhcpv4-proxy-profile)# relay information option allow-untrusted
Router(config-dhcpv4-proxy-profile)# exit
Router(config-dhcpv4)#interface PW-Ether2.201 proxy profile prox
Router(config-dhcpv4)#interface PW-Ether2.202 proxy profile prox
Router(config-dhcpv4)#commit
```

Running Configuration

This section shows DHCP relay running configuration.

```
configure
 dhcp ipv4
  profile relay1 relay
   helper-address vrf default 10.0.0.1 giaddr 0.0.0.0
   relay information option vpn
   relay information check
   relay information option
   relay information option vpn-mode rfc
   relay information option allow-untrusted
 !
 !

 interface PW-Ether2.101 relay profile relay1
 interface PW-Ether2.102 relay profile relay1
```

Verification

Verify DHCP Relay configuration.

```
Router# show dhcp ivp4 relay profile name relay1
Profile: relay1 relay
Helper Addresses:
10.0.0.1, vrf default, giaddr 0.0.0.0
Remote-Id Format : [ascii | hex]
Remote-Id value : cisco
Information Option: Enabled
Information Option Allow Untrusted: Enabled
Information Option VPN: Enabled
Information Option VPN Mode: RFC
Information Option Policy: Replace
```

Related Topics

- DHCP on PWHE Interfaces, on page 257

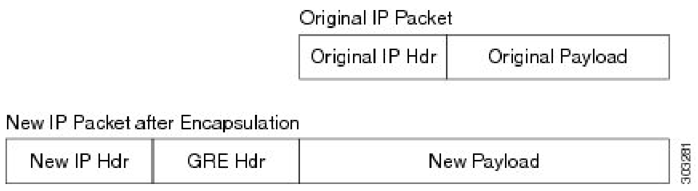
Associated Commands

- show l2vpn xconnect
- show dhcp ipv4 server binding
- show dhcp ipv4 proxy binding summary
- show dhcp ipv4 relay profile name

L2VPN over GRE

To transport an IP packet over a generic routing encapsulation (GRE) tunnel, the system first encapsulates the original IP packet with a GRE header. The encapsulated GRE packet is encapsulated once again by an outer transport header that is used to forward the packet to it’s destination. The following figure captures how GRE encapsulation over an IP transport network takes place.

Figure 33: GRE Encapsulation



Note In the new IP packet, new payload is similar to the original IP packet. Additionally, the new IP header (New IP Hdr) is similar to the tunnel IP header which in turn is similar to the transport header.

When a GRE tunnel endpoint decapsulates a GRE packet, it further forwards the packet based on the payload type. For example, if the payload is a labeled packet then the packet is forwarded based on the virtual circuit (VC) label or the VPN label for L2VPN and L3VPN respectively.

L2VPN over GRE Restrictions

Some of the restrictions that you must consider while configuring L2VPN over GRE:

- GRE over BVI is not supported.
- For VPLS flow-based load balancing scenario, the GRE tunnel is pinned down to outgoing path based on tunnel source or destination cyclic redundancy check (CRC). Unicast and flood traffic always takes the same physical path for a given GRE tunnel.
- Ingress attachment circuit must be an ASR 9000 Enhanced Ethernet Line Card for L2VPN over GRE. Additionally, GRE tunnel destination should be reachable only on an ASR 9000 Enhanced Ethernet Line Card.
- The L2VPN over GRE feature is not supported on the ASR 9000 Ethernet Line Card or Cisco ASR 9000 Series SPA Interface Processor-700 line cards as the ingress attachment circuit and GRE destination is reachable over GRE.
- Pseudowire over TE over GRE scenario is not supported.
- Preferred Path Limitations:
    - When you configure GRE as a preferred path, egress features are not supported under the GRE tunnel (Egress ACL).
    - VCCV ping or traceroute are not supported for preferred path.
    - Preferred path is supported only for pseudowires configured in a provider egde (PE) to PE topology.

GRE Deployment Scenarios

In an L2VPN network, you can deploy GRE in the following scenarios:

- Configuring GRE tunnel between provider edge (PE) to PE routers
- Configuring GRE tunnel between P to P routers
- Configuring GRE tunnel between P to PE routers

The following diagrams depict the various scenarios:

Figure 34: GRE tunnel configured between PE to PE routers

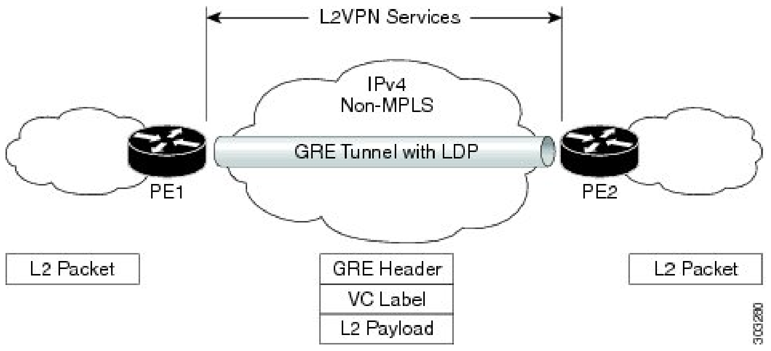


Figure 35: GRE tunnel configured between P to P routers

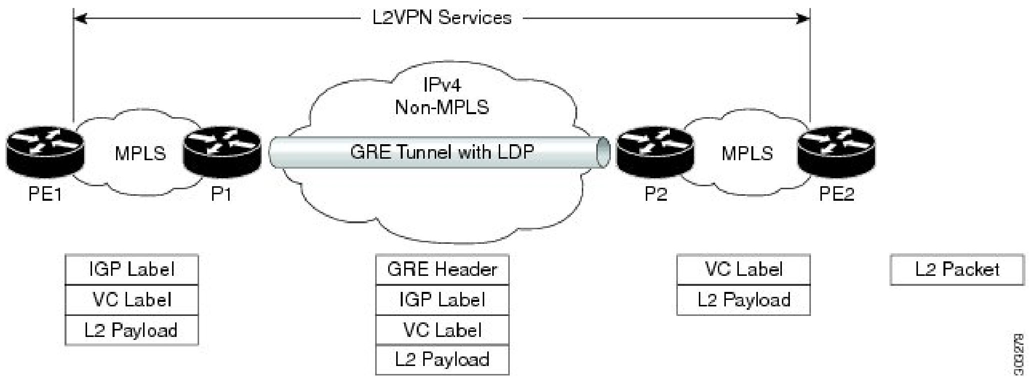
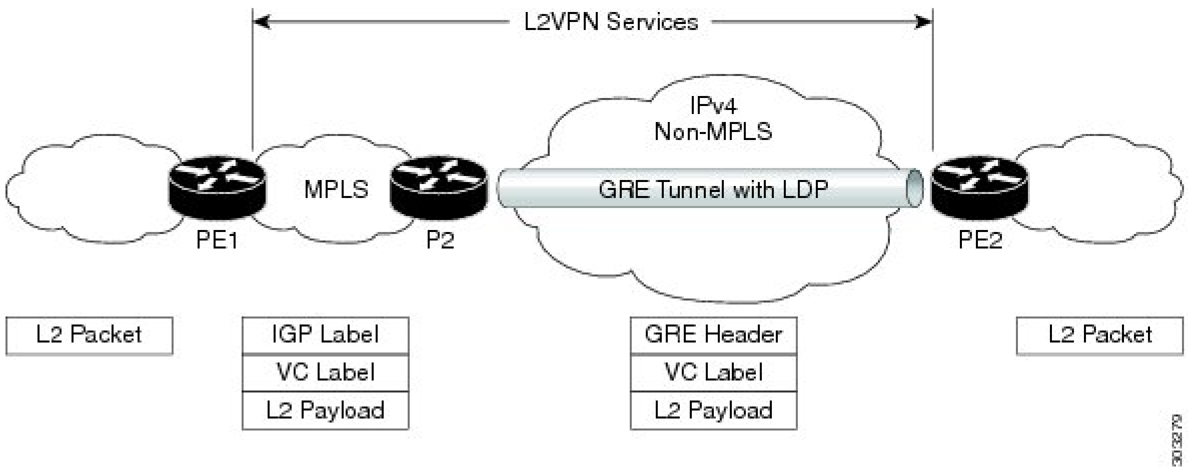


Figure 36: GRE tunnel configured between P to PE routers



Note These deployment scenarios are applicable to VPWS and VPLS.

GRE Tunnel as Preferred Path

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

How to Implement Multipoint Layer 2 Services

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

Configuring a Bridge Domain

These topics describe how to configure a bridge domain:

Creating a Bridge Domain

Perform this task to create a bridge domain .

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

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

end - Prompts user to take one of these actions:

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

Configuring a Pseudowire

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi-name* }
6. **exit**
7. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
8. **dhcp ipv4 snoop profile** *{ dhcp_snoop_profile_name }*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures the virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

- Use the *vfi-name* argument to configure the name of the specified virtual forwarding interface.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

Exits the current configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#
```

Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 8 **dhcp ipv4 snoop profile** *{ dhcp_snoop_profile_name }*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#dhcp ipv4 snoop profile profile1
```

Enables DHCP snooping on the bridge, and attaches a DHCP snooping profile.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Associating Members with a Bridge Domain

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

- Ethernet and VLAN
- VFI

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **interface** *type interface-path-id*
6. *(Optional)* **static-mac-address** { *MAC-address* }
7. **routed interface** *BVI-id*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# static-mac-address 1.1.1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 7 **routed interface** *BVI-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# routed interface BVI100
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Perform this step if you need the VPLS pseudowire traffic routed over an Integrated Routing and Bridging (IRB). This command enters bridge-group virtual interface configuration mode and adds a bridge-group virtual interface (BVI) to the bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain. This step is essential to bring the BVI's status up.

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

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

end - Prompts user to take one of these actions:

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

Configuring Bridge Domain Parameters

To configure bridge domain parameters, associate these parameters with a bridge domain:

- Maximum transmission unit (MTU)—Specifies that all members of a bridge domain have the same MTU. The bridge domain member with a different MTU size is not used by the bridge domain even though it is still associated with a bridge domain.
- Flooding—Flooding is enabled always.
- Dynamic ARP Inspection (DAI)—Ensures only valid ARP requests and responses are relayed.
- IP SourceGuard (IPSG)—Enables source IP address filtering on a Layer 2 port.

Note To verify if the DAI and IPSG features are working correctly, look up the packets dropped statistics for DAI and IPSG violation. The packet drops statistics can be viewed in the output of the **show l2vpn bridge-domain** *bd-name* <> **detail** command.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **flooding disable**
6. **mtu** *bytes*
7. **dynamic-arp-inspection** { **address-validation** | **disable** | **logging** }
8. **ip-source-guard logging**
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the l2vpn configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **flooding disable**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# flooding disable
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

Disables flooding.

Step 6 **mtu** *bytes*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mtu 1000
```

Adjusts the maximum packet size or maximum transmission unit (MTU) size for the bridge domain.

- Use the *bytes* argument to specify the MTU size, in bytes. The range is from 64 to 65535.

Step 7 **dynamic-arp-inspection** { **address-validation** | **disable** | **logging** }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# dynamic-arp-inspection
```

Enters the dynamic ARP inspection configuration submode. Ensures only valid ARP requests and responses are relayed.

Note You can configure dynamic ARP inspection under the bridge domain or the bridge port.

Step 8 **ip-source-guard logging**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# ip-source-guard logging
```

Enters the IP source guard configuration submode and enables source IP address filtering on a Layer 2 port.

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

By default, IP source guard is disabled on the bridges.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Disabling a Bridge Domain

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# shutdown
```

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

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

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

end - Prompts user to take one of these actions:

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

Blocking Unknown Unicast Flooding

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

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

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **flooding unknown-unicast disable**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
flooding unknown-unicast disable
```

Disables flooding of unknown unicast traffic at the bridge domain level.

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

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

end - Prompts user to take one of these actions:

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

Changing the Flood Optimization Mode

Perform this task to change the flood optimization mode under the bridge domain:

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **flood mode convergence-optimized**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **flood mode convergence-optimized**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# flood mode convergence-optimized
```

Changes the default flood optimization mode from Bandwidth Optimization Mode to Convergence Mode.

When you configure **flood mode convergence-optimized**, you must remove and reconfigure the bridge domain when you add, modify, or remove the pseudowire configuration of a specific bridge domain.

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

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

end - Prompts user to take one of these actions:

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

Configuring Layer 2 Security

These topics describe how to configure Layer 2 security:

Enabling Layer 2 Security

Perform this task to enable Layer 2 port security on a bridge.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge domain** *bridge-domain-name*
5. **security**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

Assigns each network interface to a bridge group and enters L2VPN bridge group configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **security**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# security
```

Enables Layer 2 port security on a bridge.

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

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

end - Prompts user to take one of these actions:

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

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

Attaching a Dynamic Host Configuration Protocol Profile

Perform this task to enable DHCP snooping on a bridge and to attach a DHCP snooping profile to a bridge.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **dhcp ipv4 snoop** { **profile** *profile-name* }
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

Assigns each network interface to a bridge group and enters L2VPN bridge group configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **dhcp ipv4 snoop** { **profile** *profile-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# dhcp ipv4 snoop profile attach
```

Enables DHCP snooping on a bridge and attaches DHCP snooping profile to the bridge.

- Use the profile keyword to attach a DHCP profile. The profile-name argument is the profile name for DHCPv4 snooping.

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

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

end - Prompts user to take one of these actions:

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

Configuring a Layer 2 Virtual Forwarding Instance

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

Creating the Virtual Forwarding Instance

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

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

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

end - Prompts user to take one of these actions:

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

Associating Pseudowires with the Virtual Forwarding Instance

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

SUMMARY STEPS

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

6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
```

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

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

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

end - Prompts user to take one of these actions:

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

Associating a Virtual Forwarding Instance to a Bridge Domain

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. **static-mac-address** { *MAC-address* }
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 **static-mac-address** { *MAC-address* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# static-mac-address 1.1.1
```

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

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

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

end - Prompts user to take one of these actions:

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

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

Attaching Pseudowire Classes to Pseudowires

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
```

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 **pw-class** { *class-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada
```

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

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

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

end - Prompts user to take one of these actions:

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

Configuring Pseudowires Using Static Labels

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**

3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **vfi** { *vfi-name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. **mpls static label** { **local** *value* } { **remote** *value* }
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
```

Adds a pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 **mpls static label** { **local** *value* } { **remote** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# mpls static label local 800 remote 500
```

Configures the MPLS static labels and the static labels for the pseudowire configuration. You can set the local and remote pseudowire labels.

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

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

end - Prompts user to take one of these actions:

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

Disabling a Virtual Forwarding Instance

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi-name* }
6. **shutdown**
7. Use the **commit** or **end** command.
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 6 **shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown
```

Disables the virtual forwarding interface (VFI).

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

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

end - Prompts user to take one of these actions:

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

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

Example:

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

Displays the state of the VFI. For example, if you shut down the VFI, the VFI is shown as shut down under the bridge domain.

Configuring the MAC Address-related Parameters

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

The MAC table attributes are set for the bridge domains.

Configuring the MAC Address Source-based Learning

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

Step 4 **bridge-domain** *bridge-domainname*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **mac**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **learning disable**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)# learning disable
```

Disables MAC learning at the bridge domain level.

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

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

end - Prompts user to take one of these actions:

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

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

Example:

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

Displays the details that the MAC address source-based learning is disabled on the bridge.

Enabling the MAC Address Withdrawal

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN l2vpn bridge group bridge domain configuration mode.

Step 5 **mac**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN l2vpn bridge group bridge domain MAC configuration mode.

Step 6 **withdrawal**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)# withdrawal
```

Enables the MAC address withdrawal for a specified bridge domain.

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

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

end - Prompts user to take one of these actions:

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

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

Example:

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

Displays detailed sample output to specify that the MAC address withdrawal is enabled. In addition, the sample output displays the number of MAC withdrawal messages that are sent over or received from the pseudowire.

Configuring the MAC Address Limit

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

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

SUMMARY STEPS

1. **configure**

2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. *(Optional)* **interface** *type interface_id*
6. **mac**
7. **limit**
8. **maximum** { *value* }
9. **action** { **flood** | **no-flood** | **shutdown** }
10. **notification** { **both** | **none** | **trap** }
11. **mac limit threshold** *80*
12. Use the **commit** or **end** command.
13. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface gigabitEthernet 0/2/0/1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

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

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

Step 6 **mac**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 7 **limit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)# limit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)#
```

Sets the MAC address limit for action, maximum, and notification and enters L2VPN bridge group bridge domain MAC limit configuration mode.

Step 8 **maximum** { *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# maximum 5000
```

Configures the specified action when the number of MAC addresses learned on a bridge is reached.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# action flood
```

Configures the bridge behavior when the number of learned MAC addresses exceed the MAC limit configured.

Step 10 **notification** { **both** | **none** | **trap** }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# notification both
```

Specifies the type of notification that is sent when the number of learned MAC addresses exceeds the configured limit.

Step 11 **mac limit threshold** *80*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# mac limit threshold 80
```

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

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

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

end - Prompts user to take one of these actions:

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

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

Example:

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

Displays the details about the MAC address limit.

Configuring the MAC Address Aging

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **mac**
6. **aging**
7. **time** { *seconds* }
8. Use the **commit** or **end** command.
9. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters the L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **mac**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6 **aging**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)# aging
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#
```

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

The maximum MAC age for ASR 9000 Ethernet and ASR 9000 Enhanced Ethernet line cards is two hours.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# time 300
```

Configures the maximum aging time.

- Use the *seconds* argument to specify the maximum age of the MAC address table entry. The range is from 300 to 30000 seconds. Aging time is counted from the last time that the switch saw the MAC address. The default value is 300 seconds.

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

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

end - Prompts user to take one of these actions:

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

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

Example:

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

Displays the details about the aging fields.

Disabling MAC Flush at the Bridge Port Level

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

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **mac**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)#
```

Enters l2vpn bridge group bridge domain MAC configuration mode.

Step 6 **port-down flush disable**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)#
port-down flush disable
```

Disables MAC flush when the bridge port becomes nonfunctional.

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

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

end - Prompts user to take one of these actions:

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

Configuring MAC Address Security

Perform this task to configure MAC address security.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group***bridge-group-name*
4. **bridge-domain***bridge-domain-name*
5. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
6. **mac**
7. **secure [action | disable | logging]**
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters L2VPN l2vpn bridge group bridge domain configuration mode.

Step 5 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000
```

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#
```

Adds an access pseudowire port to a bridge domain, or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the A.B.C.D argument to specify the IP address of the cross-connect peer.
- Use the pw-id keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 6 **mac**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)# mac
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw-mac)#
```

Enters L2VPN l2vpn bridge group bridge domain MAC configuration mode.

Step 7 **secure [action | disable | logging]**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw-mac)#
secure
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw-mac-
secure)#
```

Enters MAC secure configuration mode.

By default, bridge ports (interfaces and access pseudowires) under a bridge inherit the security configuration from the parent bridge.

Note Once a bridge port goes down, a **clear** command must be issued to bring the bridge port up.

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

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

end - Prompts user to take one of these actions:

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

Configuring an Attachment Circuit to the AC Split Horizon Group

These steps show how to add an interface to the split horizon group for attachment circuits (ACs) under a bridge domain.

SUMMARY STEPS

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

4. **bridge-domain** *bridge-domain-name*
5. **interface** *type instance*
6. **split-horizon group**
7. **commit**
8. **end**
9. **show l2vpn bridge-domain detail**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group metroA
```

Enters configuration mode for the named bridge group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain east
```

Enters configuration mode for the named bridge domain.

Step 5 **interface** *type instance*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/1/0/6
```

Enters configuration mode for the named interface.

Step 6 **split-horizon group**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# split-horizon group
```

Enters configuration mode for the named interface.

Step 7 **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# commit
```

Saves configuration changes

Step 8 **end**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# end
```

Returns to EXEC mode.

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

Example:

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

Displays information about bridges, including whether each AC is in the AC split horizon group or not.

Adding an Access Pseudowire to the AC Split Horizon Group

These steps show how to add an access pseudowire as a member to the split horizon group for attachment circuits (ACs) under a bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*
6. **split-horizon group**
7. **commit**
8. **end**
9. **show l2vpn bridge-domain detail**

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group metroA
```

Enters configuration mode for the named bridge group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain east
```

Enters configuration mode for the named bridge domain.

Step 5 **neighbor** *A.B.C.D* **pw-id** *pseudowire-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.2.2.2 pw-id 2000
```

Configures the pseudowire segment.

Step 6 **split-horizon group**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# split-horizon group
```

Adds this interface to the split horizon group for ACs. Only one split horizon group for ACs

Note Only one split horizon group for ACs and access pseudowires per bridge domain is supported

.

Step 7 **commit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw) commit
```

Saves configuration changes

Step 8 **end**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# end
```

Returns to EXEC mode.

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

Example:

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

Displays information about bridges, including whether each AC is in the AC split horizon group or not.

Configuring VPLS with BGP Autodiscovery and Signaling

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group metroA
```

Enters configuration mode for the named bridge group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain east
```

Enters configuration mode for the named bridge domain.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east
```

Enters virtual forwarding instance (VFI) configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# vpn-id 100
```

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

Step 7 **autodiscovery bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
```

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# rd auto
```

Specifies the route distinguisher (RD) under the VFI.

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target 500:99
```

Specifies the route target (RT) for the VFI.

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target import 200:20
```

Specifies the import route target for the VFI.

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target export 100:10
```

Specifies the export route target for the VFI.

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

Step 12 **signaling-protocol bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# signaling-protocol bgp
```

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# ve-id 10
```

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# ve-range 40
```

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

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

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

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

end - Prompts user to take one of these actions:

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

Configuring VPLS with BGP Autodiscovery and LDP Signaling

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# router-id 10.0.0.1
```

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

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group metroA
```

Enters configuration mode for the named bridge group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain east
```

Enters configuration mode for the named bridge domain.

Step 6 **transport-mode vlan passthrough**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# transport-mode vlan passthrough
```

Enables VC type 4 for BGP autodiscovery.

Step 7 **vfi** {*vfi-name*}

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east
```

Enters virtual forwarding instance (VFI) configuration mode.

Step 8 **autodiscovery bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
```

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

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

Step 9 **vpn-id** *vpn-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# vpn-id 100
```

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

Step 10 **rd** {*as-number:nn* | *ip-address:nn* | **auto**}

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# rd auto
```

Specifies the route distinguisher (RD) under the VFI.

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target 500:99
```

Specifies the route target (RT) for the VFI.

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target import 200:20
```

Specifies the import route target for the VFI.

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

Step 13 **route-target export** {*as-number:nn* | *ip-address:nn*}

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-target export 100:10
```

Specifies the export route target for the VFI.

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

Step 14 **signaling-protocol ldp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# signaling-protocol ldp
```

Enables LDP signaling.

Step 15 **vpls-id** {*as-number:nn* | *ip-address:nn*}

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# vpls-id 10:20
```

Specifies VPLS ID which identifies the VPLS domain during signaling.

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

Step 16 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring Service Path Preference

Perform these tasks to configure Service Path Preference:

Setting a Forward Class in a Route Policy

The following configuration shows how to set a forward-class in a route-policy:

```
route-policy  fwd1
      set forward-class 1
end-policy
!
route-policy fwd2
      set forward-class 2
 end-policy
!
```

Attaching a Route Policy at Table Policy Attach Point

The following configuration shows how to attach a route-policy to a table-policy attach point for VPLS bridge-domain VFI:

```
config
  l2vpn
    bridge group bg1
     bridge-domain bd1
      vfi v1
       autodiscovery bgp
        table-policy fwd1
!
```

The following configuration shows how to attach a route-policy to a table-policy attach point for EVPN EVI:

```
config
  l2vpn
    bridge group pbb
     bridge-domain core1
      pbb core
       evi 1
!
 bridge group edge
  bridge-domain edge1
   pbb edge i-sid 256 core-bridge core1
!
 evpn
  evi 1
    bgp
     table-policy fwd2
!
```

Associating a TE Tunnel with Forward Class Index

The following configuration shows how to associate a TE tunnel with forward-class index:

```
 config
  interface tunnel-te1
   ipv4 unnumbered Loopback0
   autoroute announce
   destination 10.10.10.10
   forward-class 1
   path-option 10 explicit name PATH1
!
```

Enabling Route Policy for L2VPN VPLS with BGP Autodiscovery

Perform this task to enable route-policy for L2VPN VPLS with BGP autodiscovery configuration. Only route-policy export is supported.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **vfi** { *vfi-name* }
6. **autodiscovery bgp**
7. **route-policy export** *policy-name*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
```

Enters configuration mode for the named bridge group.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
```

Enters configuration mode for the named bridge domain.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east
```

Enters virtual forwarding instance (VFI) configuration mode.

Step 6 **autodiscovery bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
```

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

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

Step 7 **route-policy export** *policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# route-policy export RPL_1
```

Attaches the route-policy at **route-policy export** attach point.

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

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

end - Prompts user to take one of these actions:

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

Example

Enabling Route Policy for L2VPN VPWS with BGP Autodiscovery

Perform this task to enable route-policy for L2VPN VPWS with BGP autodiscovery configuration. Only route-policy export is supported.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *xconnect group name*
4. **mp2mp** *mp2mp instance name*
5. **autodiscovery bgp**
6. **route-policy export** *policy-name*

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group xg1
```

Enters configuration mode for the named cross-connect group.

Step 4 **mp2mp** *mp2mp instance name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# mp2mp mp1
```

Creates named mp2mp instance.

Step 5 **autodiscovery bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-mp2mp)# autodiscovery bgp
```

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

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

Step 6 **route-policy export** *policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-mp2mp-ad)# route-policy export RPL_2
```

Attaches the route-policy at **route-policy export** attach point.

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

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

end - Prompts user to take one of these actions:

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

Example

Configuring G.8032 Ethernet Ring Protection

To configure the G.8032 operation, separately configure:

- An ERP instance to indicate:
  - which (sub)interface is used as the APS channel
  - which (sub)interface is monitored by CFM
  - whether the interface is an RPL link, and, if it is, the RPL node type
- CFM with EFD to monitor the ring links

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 ERP instance, if different from default values. This is optional.

This section provides information on:

Configuring ERP Profile

Perform this task to configure Ethernet ring protection (ERP) profile.

SUMMARY STEPS

1. **configure**
2. **Ethernet ring g8032 profile** *profile-name*
3. **timer** { **wtr** | **guard** | **hold-off** } *seconds*
4. **non-revertive**
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **Ethernet ring g8032 profile** *profile-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# Ethernet ring g8032 profile p1
```

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

Step 3 **timer** { **wtr** | **guard** | **hold-off** } *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-g8032-ring-profile)# timer hold-off 5
```

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

Step 4 **non-revertive**

Example:

```
RP/0/RSP0/CPU0:router(config-g8032-ring-profile)# non-revertive
```

Specifies a non-revertive ring instance.

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

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

end - Prompts user to take one of these actions:

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

Configuring CFM MEP

For more information about Ethernet Connectivity Fault Management (CFM), refer to the *Configuring Ethernet OAM on the Cisco ASR 9000 Series Router* module in the Cisco *ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide*.

Configuring an ERP Instance

Perform this task to configure an ERP instance.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **interface** *type port0-interface-path-id.subinterface*
6. **interface** *type port1-interface-path-id.subinterface*
7. **bridge-domain** *domain-name*
8. **interface** *type interface-path-id.subinterface*
9. **ethernet ring g8032** *ring-name*
10. **instance** *number*
11. **description** *string*
12. **profile** *profile-name*
13. **rpl { port0 | port1 } { owner | neighbor | next-neighbor }**
14. **inclusion-list vlan-ids** *vlan-id*
15. **aps-channel**
16. **level** *number*
17. **port0 interface** *type path-id*
18. **port1** { **interface** *type interface-path-id* | **bridge-domain** *bridge-domain-name* | **xconnect** *xconnect-name* | **none** }
19. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 5 **interface** *type port0-interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 6 **interface** *type port1-interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 7 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd2
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

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

Step 8 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0.10
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 9 **ethernet ring g8032** *ring-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# ethernet ring g8032 r1
```

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

Step 10 **instance** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp)# instance 1
```

Enters the Ethernet ring G.8032 instance configuration submode.

Step 11 **description** *string*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance)# description test
```

Specifies a string that serves as description for that instance.

Step 12 **profile** *profile-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance)#profile p1
```

Specifies associated Ethernet ring G.8032 profile.

Step 13 **rpl { port0 | port1 } { owner | neighbor | next-neighbor }**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance)#rpl port0 neighbor
```

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

Step 14 **inclusion-list vlan-ids** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance)# inclusion-list vlan-ids e-g
```

Associates a set of VLAN IDs with the current instance.

Step 15 **aps-channel**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance)# aps-channel
```

Enters the Ethernet ring G.8032 instance aps-channel configuration submode.

Step 16 **level** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance-aps)# level 5
```

Specifies the APS message level. The range is from 0 to 7.

Step 17 **port0 interface** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(configl2vpn-erp-instance-aps)# port0 interface GigabitEthernet 0/0/0/0.1
```

Associates G.8032 APS channel interface to port0.

Step 18 **port1** { **interface** *type interface-path-id* | **bridge-domain** *bridge-domain-name* | **xconnect** *xconnect-name* | **none** }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-instance-aps)# port1 interface GigabitEthernet 0/0/0/1.1
```

Associates G.8032 APS channel interface to port1.

Step 19 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring ERP Parameters

Perform this task to configure ERP parameters.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **ethernet ring g8032** *ring-name*
4. **port0 interface** *type interface-path-id*
5. **monitor port0 interface** *type interface-path-id*
6. **exit**
7. **port1** { **interface** *type interface-path-id* | **virtual** | **none**}
8. **monitor port1 interface** *type interface-path-id*
9. **exit**
10. **exclusion-list vlan-ids** *vlan-id*
11. **open-ring**
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **ethernet ring g8032** *ring-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# ethernet ring g8032 r1
```

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

Step 4 **port0 interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp)# port0 interface GigabitEthernet 0/1/0/6
```

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

Step 5 **monitor port0 interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-port0)# monitor port0 interface 0/1/0/2
```

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 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-port0)# exit
```

Exits port0 configuration submode.

Step 7 **port1** { **interface** *type interface-path-id* | **virtual** | **none**}

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp)# port1 interface GigabitEthernet 0/1/0/8
```

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

Step 8 **monitor port1 interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-port1)# monitor port1 interface 0/1/0/3
```

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 9 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp-port1)# exit
```

Exits port1 configuration submode.

Step 10 **exclusion-list vlan-ids** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp)# exclusion-list vlan-ids a-d
```

Specifies a set of VLAN IDs that is not protected by Ethernet ring protection mechanism.

Step 11 **open-ring**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-erp)# open-ring
```

Specifies Ethernet ring G.8032 as open ring.

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

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

end - Prompts user to take one of these actions:

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

Configuring TCN Propagation

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **tcn-propagation**
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **tcn-propagation**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# tcn-propagation
```

Allows TCN propagation from minor ring to major ring and from MSTP to G.8032.

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

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

end - Prompts user to take one of these actions:

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

Configuring Flow Aware Transport Pseudowire

This section provides information on

Enabling Load Balancing with ECMP and FAT PW for VPWS

Perform this task to enable load balancing with ECMP and FAT PW for VPWS. Creating a PW-Class in L2VPN configuration leads to load-balancing.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **pw-class** { *name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class path1
```

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

Step 4 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
```

Configures the pseudowire encapsulation to MPLS.

Step 5 **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
```

Enables load-balancing on ECMPs. Also, enables the imposition and disposition of flow labels for the pseudowire.

Note If the static keyword is not specified, end to end negotiation of the FAT PW is enabled.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)#
```

Exits the pseudowire encapsulation submode and returns the router to the parent configuration mode.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)#exit
RP/0/RSP0/CPU0:router(config-l2vpn)#
```

Exits the pseudowire submode and returns the router to the l2vpn configuration mode.

Step 8 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group grp1
RP/0/RSP0/CPU0:router(config-l2vpn-xc)#
```

Specifies the name of the cross-connect group.

Step 9 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p vlan1
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#
```

Specifies the name of the point-to-point cross-connect.

Step 10 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface TenGigE 0/0/0/0.1
```

Specifies the interface type and instance.

Step 11 **neighbor** *A.B.C.D* **pw-id** pseudowire-id

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.2.2.2 pw-id 2000
```

Configures the pseudowire segment for the cross-connect.

Use the A.B.C.D argument to specify the IP address of the cross-connect peer.

Note A.B.C.D can be a recursive or non-recursive prefix.

Step 12 **pw-class** *class-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class path1
```

Associates the pseudowire class with this pseudowire.

Step 13 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

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

Enabling Load Balancing with ECMP and FAT PW for VPLS

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **load-balancing flow {src-dst-mac | src-dst-ip}**
4. **pw-class** { *class - name* }
5. **encapsulation mpls**
6. **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]
7. **exit**
8. **bridge group** *bridge-group-name*
9. **bridge-domain** *bridge-domain-name*
10. **vfi** { *vfi-name* }
11. **autodiscovery bgp**
12. **signaling-protocol bgp**
13. **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]
14. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **load-balancing flow {src-dst-mac | src-dst-ip}**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# load-balancing flow src-dst-ip
```

Enables flow based load balancing.

- **src-dst-mac**—Uses source and destination MAC addresses for hashing.
- **src-dst-ip**—Uses source and destination IP addresses for hashing.

Note It is recommended to use the **load-balancing flow** command with the **src-dst-ip** keyword.

Step 4 **pw-class** { *class - name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class class1
```

Associates the pseudowire class with this pseudowire.

Step 5 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
```

Configures the pseudowire encapsulation to MPLS.

Step 6 **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)# load-balancing flow-label both
```

Enables load-balancing on ECMPs. Also, enables the imposition and disposition of flow labels for the pseudowire.

Note If the **static** keyword is not specified, end to end negotiation of the **FAT PW** is enabled.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)# exit
```

Exits the pseudowire encapsulation submode and returns the router to the parent configuration mode.

Step 8 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group group1
```

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

Step 9 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain domain1
```

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

Step 10 **vfi** { *vfi-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi my_vfi
```

Enters virtual forwarding instance (VFI) configuration mode.

Step 11 **autodiscovery bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
```

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

Step 12 **signaling-protocol bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad)# signaling-protocol bgp
```

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

Step 13 **load-balancing flow-label** { **both** | **code** | **receive** | **transmit** } [**static**]

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# load-balancing flow-label both static
```

Enables load-balancing on ECMPs. Also, enables the imposition and disposition of flow labels for the pseudowire.

Step 14 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring Pseudowire Headend

The Pseudowire Headend (PWHE) is created by configuring the pw-ether main interface, pw-ether subinterface, or pw-iw interface. The available PWHE types are pw-ether main interfaces, subinterfaces, and pw-iw interfaces. Unless specified otherwise, the term interface is applicable for pw-ether main interfaces, subinterfaces, and pw-iw interfaces.

Note The PWHE-Ethernet subinterfaces and interworking interfaces are supported from Release 5.1.1 onwards.

For the PWHE to be functional, the crossconnect has to be configured completely. Configuring other Layer 3 (L3) parameters, such as VRF and IP addresses, are optional for the PWHE to be functional. However, the L3 features are required for the Layer 3 services to be operational; that is, for PW L3 termination.

PWHE supports both IPv4 and IPv6 addresses.

This section describes these topics:

PWHE Configuration Restrictions

These configuration restrictions are applicable for PWHE:

1. The generic interface list members must be the superset of the ECMP path list to the A-PE.
2. Only eight generic interface lists are supported per A-PE neighbor address.
3. Eight Layer 3 links per generic interface list are supported.
4. Only PW-Ether interfaces can be configured as PWHE L2 or L3 subinterfaces.
5. Cross-connects that contain PW-Ether main interfaces can be configured as either VC-type 5 or VC-type 4. By default, the cross-connects are configured as VC-type 5; else by using the command—pw-class transport-mode ethernet. To configure the cross-connects as VC-type 4, use the **p2p neighbor tag-impose vlan id** and the **pw-class transport-mode vlan** commands.
6. Cross-connects that contain PW-Ether main interfaces that have L3 PW-Ether subinterfaces associated with them, are supported with only VC-type 5.
7. Cross-connects that contain PW-IW interfaces are only supported with IPv4 and VC-type 11. PW-IW interfaces are the L3 virtual interfaces used for IP interworking. To configure the cross-connect as VC-type 11, use the interworking ipv4 command.
8. PW-Ether interfaces and subinterfaces can be configured with both IPv4 and IPv6.
9. PW-IW interfaces can be configured only with IPv4.
10. Pseudowire redundancy, preferred path, local switching or L2TP are not supported for crossconnects configured with PWHE.
11. Applications such as TE and LDP have checks for interface type and therefore do not allow PWHE to be configured.
12. Address family, CDP and MPLS configurations are not allowed on PWHE interfaces.
13. For PWHE, eBGP, static routes, OSPF, and ISIS are supported with both IPv4 and IPv6. RIP is only supported with IPv4 and not with IPv6.
14. MAC address is not supported for a pw-iw interface.
15. PW-Ether interfaces with different remote neighbors need to be attached with a different generic interface list. 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, in Figure 30: Pseudowire Network, for each A-PE that the S-PE peers with, a unique generic interface list needs to be configured. The generic interface list may have the same set of the outgoing interface(s).
16. Refer PWHE Load Balancing Support using FAT Label topic for the feature's restrictions.

Configuring Generic Interface List

Perform this task to configure a generic interface list.

Note Only eight generic interface lists are supported per A-PE neighbor address. Eight Layer 3 links per generic interface list are supported. Repeat Step 3 or Step 4 to add the interfaces to the generic interface list.

SUMMARY STEPS

1. **configure**
2. **generic-interface-list** *list-name*
3. **interface** *type interface-path-id*
4. **interface** *type interface-path-id*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **generic-interface-list** *list-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# generic-interface-list list1
```

Configures a generic interface list.

To remove the generic interface list, use the no form of the command, that is, **no generic-interface-list** *list-name*.

Step 3 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if-list)# interface Bundle-Ether 100
```

Configures the specified interface.

Step 4 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if-list)# interface Bundle-Ether 200
```

Configures the specified interface.

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

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

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.

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

Configuring PWHE Interfaces

Perform this task to configure PWHE Ethernet and interworking interfaces, that is, to attach the generic interface list with a PWHE Ethernet and interworking interfaces.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config)# interface pw-ether <id>
or
RP/0/RSP0/CPU0:router(config)# interface pw-iw <id>
```

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

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

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

Example:

```
RP/0/RSP0/CPU0:router(config-if)# attach generic-interface-list interfacelist1
```

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

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

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

end - Prompts user to take one of these actions:

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

Configuring PWHE Crossconnect

Perform this task to configure PWHE crossconnects.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface pw-ether** *id* or **interface pw-iw** *id*
6. **neighbor** *ip-address* **pw-id** *value*
7. **pw-class** *class-name*
8. (*Only PW-IW*) **interworking ipv4**
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group pw-he1
```

Configures a cross-connect group name using a free-format 32-character string.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p pw-hexconnect
```

Enters P2P configuration submode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface pw-ether 100
or
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p )# interface pw-iw 100
```

Configures the PWHE interface.

Step 6 **neighbor** *ip-address* **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.165.200.25 pw-id 100
```

Configures a pseudowire for a cross-connect.

The IP address is that of the corresponding PE node.

The **pw-id** must match the **pw-id** of the PE node.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Note The pseudowire class should be defined under l2vpn for VC4 and VC5 as follows:

```
pw-class vc_type_4
encapsulation mpls
transport-mode vlan
!
!
pw-class vc_type_5
encapsulation mpls
transport-mode ethernet
!
!
```

Step 8 (*Only PW-IW*) **interworking ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# interworking ipv4
```

Configures the cross-connect p2p entity to use VC-type 11 or the IP interworking mode in the establishment of the pseudowire.

Step 9 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring the Source Address

Perform this task to configure the local source address

Note Only IPv4 is supported as pw-class source address.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation mpls**
5. **ipv4 source** *source-address*
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters Layer 2 VPN configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class class1
```

Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 4 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
```

Configures the pseudowire encapsulation to MPLS.

Step 5 **ipv4 source** *source-address*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-mpls)# ipv4 source 10.1.1.1
```

Sets the local source IPv4 address.

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

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

end - Prompts user to take one of these actions:

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

Configuring PWHE Interface Parameters

Perform this task to configure PWHE interface parameters.

SUMMARY STEPS

1. **configure**
2. **interface pw-ether** *id* (or) **interface pw-iw** *id*
3. **ipv4 address** *ip-address subnet-mask* (or) (Only PW-Ether) **ipv6 address** *ipv6-prefix/prefix-length*
4. **attach generic-interface-list** *interface_list_name*
5. **l2overhead** *bytes*
6. **load-interval** *seconds*
7. **dampening** *decay-life*
8. **logging events link-status**
9. (Only PW-Ether main interfaces) **mac-address** *MAC address*
10. **mtu** *interface_MTU*
11. **bandwidth** *kbps*
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface pw-ether** *id* (or) **interface pw-iw** *id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface pw-ether <id>
or
RP/0/RSP0/CPU0:router(config)# interface pw-iw <id>
```

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

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

Step 3 **ipv4 address** *ip-address subnet-mask* (or) (Only PW-Ether) **ipv6 address** *ipv6-prefix/prefix-length*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 40.1.1.2 255.255.255.0
```

Sets the IPv4 or IPv6 address of the interface.

Step 4 **attach generic-interface-list** *interface_list_name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# attach generic-interface-list interfacelist1
```

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

Step 5 **l2overhead** *bytes*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# l2overhead 20
```

Sets layer 2 overhead size.

Step 6 **load-interval** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# load-interval 90
```

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

The interval:

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

Step 7 **dampening** *decay-life*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# dampening 10
```

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

Step 8 **logging events link-status**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# logging events link-status
```

Configures per interface logging.

Step 9 (Only PW-Ether main interfaces) **mac-address** *MAC address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# mac-address aaaa.bbbb.cccc
```

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

Step 10 **mtu** *interface_MTU*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# mtu 128
```

Sets the MTU on an interface.

Step 11 **bandwidth** *kbps*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# bandwidth 200
```

Configures the bandwidth. The range is from 0 to 4294967295.

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

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

end - Prompts user to take one of these actions:

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

Configuring PWHE Layer 2 Subinterfaces and Adding it to the Bridge-domain

Perform this task to configure PWHE layer 2 subinterfaces and add it to the bridge-domain.

Note A Layer 2 subinterface does not contain an IP address and must be configured to operate in the Layer 2 transport mode.

SUMMARY STEPS

1. **configure**
2. **interface pw-ether** *id*

3. **ipv4 address** *ip-address subnet-mask* (or) **ipv6 address** *ipv6-prefix/prefix-length*
4. **attach generic-interface-list** *interface_list_name*
5. **interface pw-ether** *id.subintfid* **l2transport**
6. **encapsulation dot1q** *value*
7. **l2vpn**
8. **xconnect group** *group-name*
9. **p2p** *xconnect-name*
10. **interface pw-ether** *id*
11. **neighbor ipv4** *ip-address* **pw-id** *value*
12. **bridge group** *bridge-group-name*
13. **bridge-domain** *bridge-domain-name*
14. **interface pw-ether** *id.subintfid*
15. **interface** *type interface-path-id*
16. **neighbor** *ip-address* **pw-id** *value*
17. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

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

Example:

```
RP/0/RSP0/CPU0:router(config)# interface PW-Ether1
```

Configures the PWHE interface and enters the interface configuration mode.

Step 3 **ipv4 address** *ip-address subnet-mask* (or) **ipv6 address** *ipv6-prefix/prefix-length*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 40.1.1.2 255.255.255.0
```

Sets the IPv4 or IPv6 address of the main interface.

Step 4 **attach generic-interface-list** *interface_list_name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# attach generic-interface-list pw_he
```

Attaches the generic interface list to the interface.

Step 5 **interface pw-ether** *id.subintfid* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# interface PW-Ether1.1 l2transport
```

Configures the PWHE sub-interface and enters the sub-interface configuration mode.

Step 6 **encapsulation dot1q** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 1
```

Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.

Step 7 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 8 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group xg
```

Configures a cross-connect group name using a free-format 32-character string.

Step 9 **p2p** *xconnect-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p1
```

Enters P2P configuration submode.

Step 10 **interface pw-ether** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface PW-Ether1
```

Configures the PWHE interface.

Step 11 **neighbor ipv4** *ip-address* **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor ipv4 1.1.1.1 pw-id 1
```

Configures a pseudowire for a cross-connect.

The IP address is that of the corresponding A-PE node.

The pw-id must match the pw-id of the A-PE node.

Step 12 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p-pw)# bridge group bg
```

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

Step 13 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 14 **interface pw-ether** *id.subintfid*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface PW-Ether1.1
```

Adds the subinterface to the bridge domain.

Step 15 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# interface GigabitEthernet0/1/1/11.1
```

Enters interface configuration mode and adds a subinterface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

Step 16 **neighbor** *ip-address* **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# neighbor 3.3.3.3 pw-id 101
```

Configures a pseudowire for a bridge-domain.

Step 17 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring PWHE Layer 3 Subinterfaces

Perform this task to configure PWHE layer 3 subinterfaces.

Note A layer 3 subinterface must have an IPv4 or IPv6 address and cannot be configured in the layer 2 transport mode.

SUMMARY STEPS

1. **configure**
2. **interface pw-ether** *id*
3. **ipv4 address** *ip-address subnet-mask* (or) **ipv6 address** *ipv6-prefix/prefix-length*
4. **attach generic-interface-list** *interface_list_name*
5. **interface pw-ether** *id.subintfid*
6. **ipv4 address** *ip-address subnet-mask* (or) **ipv6 address** *ipv6-prefix/prefix-length*
7. **encapsulation dot1q** *value*
8. **l2vpn**
9. **xconnect group** *group-name*
10. **p2p** *group-name*
11. **interface pw-ether** *id*
12. **neighbor ipv4** *ip-address* **pw-id** *value*

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface pw-ether** *id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface PW-Ether1
```

Configures the PWHE interface and enters the interface configuration mode.

Step 3 **ipv4 address** *ip-address subnet-mask* (or) **ipv6 address** *ipv6-prefix/prefix-length*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 40.1.1.2 255.255.255.0
```

Sets the IPv4 or the IPv6 address of the main interface.

Step 4 **attach generic-interface-list** *interface_list_name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# attach generic-interface-list pw_he
```

Attaches the generic interface list to the interface.

Step 5 **interface pw-ether** *id.subintfid*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# interface PW-Ether1.1
```

Configures the PWHE sub-interface and enters the sub-interface configuration mode.

Step 6 **ipv4 address** *ip-address subnet-mask* (or) **ipv6 address** *ipv6-prefix/prefix-length*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 40.1.1.2 255.255.255.0
```

Sets the IPv4 or the IPv6 address of the subinterface.

Step 7 **encapsulation dot1q** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 1
```

Defines the matching criteria to map 802.1Q frames ingress on an interface to the appropriate service instance.

Step 8 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# l2vpn
```

Enters Layer 2 VPN configuration mode.

Step 9 **xconnect group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# xconnect group xg
```

Configures a cross-connect group name using a free-format 32-character string.

Step 10 **p2p** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc)# p2p 1
```

Enters P2P configuration submode.

Step 11 **interface pw-ether** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# interface PW-Ether1
```

Configures the PWHE interface.

Step 12 **neighbor ipv4** *ip-address* **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor ipv4 1.1.1.1 pw-id 1
```

Configures a pseudowire for a cross-connect.

The IP address is that of the corresponding A-PE node.

The **pw-id** must match the **pw-id** of the A-PE node.

Configuring L2VPN over GRE

Perform these tasks to configure L2VPN over GRE.

SUMMARY STEPS

1. **configure**
2. **interface type** *interface-path-id*
3. **l2transport**
4. **exit**
5. **interface loopback instance**
6. **ipv4 address ip-address**
7. **exit**
8. **interface loopback** *instance*
9. **ipv4 address** *ip-address*
10. **router ospf** *process-name*
11. **area** *area-id*
12. **interface loopback** *instance*
13. **interface tunnel-ip** *number*
14. **exit**
15. **interface tunnel-ip** *number*
16. **ipv4 address** *ipv4-address subnet-mask*
17. **tunnel source** *type path-id*
18. **tunnel destination** *ip-address*
19. **end**
20. **l2vpn**
21. **bridge group** *bridge-group-name*
22. **bridge-domain** *bridge-domain-name*

23. **interface type** *interface-path-id*
24. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
25. **mpls ldp**
26. **router-id** { *router-id* }
27. **interface tunnel-ip** *number*
28. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router# interface TenGigE0/1/0/12
```

Enters interface configuration mode and configures an interface.

Step 3 **l2transport**

Example:

```
RP/0/RSP0/CPU0:router# l2transport
```

Enables Layer 2 transport on the selected interface.

Step 4 **exit**

Example:

```
RP/0/RSP0/CPU0:router# exit
```

Exits the current configuration mode.

Step 5 **interface loopback instance**

Example:

```
RP/0/RSP0/CPU0:router# interface Loopback0
```

Enters interface configuration mode and names the new loopback interface.

Step 6 **ipv4 address ip-address**

Example:

```
RP/0/RSP0/CPU0:router# ipv4 address 100.100.100.100 255.255.255.255
```

Assigns an IP address and subnet mask to the virtual loopback interface.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router# exit
```

Exits the current configuration mode.

Step 8 **interface loopback** *instance*

Example:

```
RP/0/RSP0/CPU0:router# interface Loopback1
```

Enters interface configuration mode and names the new loopback interface.

Step 9 **ipv4 address** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router# ipv4 address 10.0.1.1 255.255.255.255
```

Assigns an IP address and subnet mask to the virtual loopback interface.

Step 10 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router# router ospf 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Step 11 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

Step 12 **interface loopback** *instance*

Example:

```
RP/0/RSP0/CPU0:router# interface Loopback0
```

Enters interface configuration mode and names the new loopback interface.

Step 13 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router# interface tunnel-ip1
```

Enters tunnel interface configuration mode.

Step 14 **exit**

Example:

```
RP/0/RSP0/CPU0:router# exit
```

Exits the current configuration mode.

Step 15 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip1
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 16 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 12.0.0.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 17 **tunnel source** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source Loopback1
```

Specifies the source of the tunnel interface.

Step 18 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 100.100.100.20
```

Defines the tunnel destination.

Step 19 **end**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# end
```

Saves configuration changes.

- When you issue the **end** command, the system prompts you to commit changes:

  ```
  Uncommitted changes found, commit them before exiting(yes/no/cancel)?[cancel]:
  ```

  - Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to **EXEC** mode.
  - Entering **no** exits the configuration session and returns the router to **EXEC** mode without committing the configuration changes.
  - Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

Step 20 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router# l2vpn
```

Enters L2VPN configuration mode.

Step 21 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router# bridge group access-pw
```

Creates a bridge group that can contain bridge domains, and then assigns network interfaces to the bridge domain.

Step 22 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router# bridge-domain test
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 23 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router# interface TenGigE0/1/0/12
```

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 24 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router# neighbor 125.125.125.125 pw-id 100
```

Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the A.B.C.D argument to specify the IP address of the cross-connect peer

  Note A.B.C.D can be a recursive or non-recursive prefix.

- Use the pw-id keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 25 **mpls ldp**

Example:

```
RP/0/RSP0/CPU0:router# mpls ldp
```

Enables MPLS LDP configuration mode.

Step 26 **router-id** { *router-id* }

Example:

```
RP/0/RSP0/CPU0:router# router-id 100.100.100.100
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 27 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router# interface tunnel-ip1
```

Enters tunnel interface configuration mode.

Note The number argument refers to the number associated with the tunnel interface

Step 28 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a GRE Tunnel as Preferred Path for Pseudowire

Perform this task to configure a GRE tunnel as the preferred path for pseudowires.

SUMMARY STEPS

1. **configure**
2. **l2vpn**

3. **pw-class** { *name* }
4. **encapsulation mpls**
5. **preferred-path** { **interface** } { **tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value* } [**fallback disable**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **pw-class** { *name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class gre
```

Configures the pseudowire class name.

Step 4 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
```

Configures the pseudowire encapsulation to MPLS.

Step 5 **preferred-path** { **interface** } { **tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value* } [**fallback disable**]

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-encap-
mpls)# preferred-path interface tunnel-ip 1 fallback disable
```

Configures preferred path tunnel settings. If the fallback disable configuration is used and once the TE/TP tunnel is configured as the preferred path goes down, the corresponding pseudowire can also go down.

Note Ensure that fallback is supported.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Examples for Multipoint Layer 2 Services

This section includes these configuration examples:

Multipoint Layer 2 Services Configuration for Provider Edge-to-Provider Edge: Example

These configuration examples show how to create a Layer 2 VFI with a full-mesh of participating Multipoint Layer 2 Services provider edge (PE) nodes.

This configuration example shows how to configure PE 1:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE1-VPLS-A
    interface TenGigE0/0/0/0
    vfi 1
     neighbor 172.16.0.1 pw-id 1
     neighbor 192.168.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 10.0.0.1 255.0.0.0
```

This configuration example shows how to configure PE 2:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE2-VPLS-A
    interface TenGigE0/0/0/1

    vfi 1
     neighbor 10.0.0.1 pw-id 1
     neighbor 192.168.0.1 pw-id 1
     !
   !
 interface loopback 0
  ipv4 address 172.16.0.1 255.240.0.0
```

This configuration example shows how to configure PE 3:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE3-VPLS-A
    interface TenGigE0/0/0/2
```

```
      vfi 1
       neighbor 10.0.0.1 pw-id 1
       neighbor 172.16.0.1 pw-id 1
       !
    !
   interface loopback 0
    ipv4 address 192.168.0.1 255.255.0.0
```

Multipoint Layer 2 Services Configuration for Provider Edge-to-Customer Edge: Example

This configuration shows how to configure Multipoint Layer 2 Services for a PE-to-CE nodes:

```
configure
 interface TenGigE0/0/0/0
  l2transport---AC interface

  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
```

Displaying MAC Address Withdrawal Fields: Example

This sample output shows the MAC address withdrawal fields:

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain detail

Legend: pp = Partially Programmed.
Bridge group: 222, bridge-domain: 222, id: 0, state: up, ShgId: 0, MSTi: 0
  Coupled state: disabled
  MAC learning: enabled
  MAC withdraw: enabled
    MAC withdraw sent on: bridge port up
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no
  MAC port down flush: enabled
  MAC Secure: disabled, Logging: disabled
  Split Horizon Group: none
  Dynamic ARP Inspection: disabled, Logging: disabled
  IP Source Guard: disabled, Logging: disabled
  DHCPv4 snooping: disabled
  IGMP Snooping: enabled
  IGMP Snooping profile: none
  MLD Snooping profile: none
  Storm Control: disabled
  Bridge MTU: 1500
  MIB cvplsConfigIndex: 1
  Filter MAC addresses:
  P2MP PW: disabled
  Create time: 01/03/2017 11:01:11 (00:21:33 ago)
  No status change since creation
  ACs: 1 (1 up), VFIs: 1, PWs: 1 (1 up), PBBs: 0 (0 up)
  List of ACs:
    AC: TenGigE0/2/0/1.7, state is up
      Type VLAN; Num Ranges: 1
      Outer Tag: 21
      VLAN ranges: [22, 22]
      MTU 1508; XC ID 0x208000b; interworking none
      MAC learning: enabled
      Flooding:
        Broadcast & Multicast: enabled
        Unknown unicast: enabled
      MAC aging time: 300 s, Type: inactivity
      MAC limit: 4000, Action: none, Notification: syslog
      MAC limit reached: no
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      Dynamic ARP Inspection: disabled, Logging: disabled
      IP Source Guard: disabled, Logging: disabled
      DHCPv4 snooping: disabled
      IGMP Snooping: enabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
      Static MAC addresses:
      Statistics:
        packets: received 714472608 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 97708776
        bytes: received 88594603392 (multicast 0, broadcast 0, unknown unicast 0, unicast
0), sent 12115888224
        MAC move: 0
      Storm control drop counters:
        packets: broadcast 0, multicast 0, unknown unicast 0
        bytes: broadcast 0, multicast 0, unknown unicast 0
      Dynamic ARP inspection drop counters:
        packets: 0, bytes: 0
      IP source guard drop counters:
        packets: 0, bytes: 0
  List of VFIs:
    VFI 222 (up)
      PW: neighbor 10.0.0.1, PW ID 222, state is up ( established )
        PW class not set, XC ID 0xc000000a
        Encapsulation MPLS, protocol LDP
        Source address 21.21.21.21
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ -------------------------
          Label        24017                          24010
          Group ID     0x0                            0x0
          Interface    222                            222
          MTU          1500                           1500
          Control word disabled                       disabled
          PW type      Ethernet                       Ethernet
          VCCV CV type 0x2                            0x2
                       (LSP ping verification)        (LSP ping verification)
          VCCV CC type 0x6                            0x6
                       (router alert label)           (router alert label)
                       (TTL expiry)                   (TTL expiry)
          ------------ ------------------------------ -------------------------
        Incoming Status (PW Status TLV):
          Status code: 0x0 (Up) in Notification message
        MIB cpwVcIndex: 3221225482
        Create time: 01/03/2017 11:01:11 (00:21:33 ago)
        Last time status changed: 01/03/2017 11:21:01 (00:01:43 ago)
        Last time PW went down: 01/03/2017 11:15:21 (00:07:23 ago)
        MAC withdraw messages: sent 0, received 0
        Forward-class: 0
        Static MAC addresses:
        Statistics:
          packets: received 95320440 (unicast 0), sent 425092569
          bytes: received 11819734560 (unicast 0), sent 52711478556
          MAC move: 0
        Storm control drop counters:
          packets: broadcast 0, multicast 0, unknown unicast 0
          bytes: broadcast 0, multicast 0, unknown unicast 0
      DHCPv4 snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      VFI Statistics:
        drops: illegal VLAN 0, illegal length 0
```

Split Horizon Group: Example

This example configures interfaces for Layer 2 transport, adds them to a bridge domain, and assigns them to split horizon groups.

```
RP/0/RSP0/CPU0:router(config)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain all_three
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet 0/0/0/0.99
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet 0/0/0/0.101
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#split-horizon group
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#neighbor 192.168.99.1 pw-id 1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#neighbor 192.168.99.9 pw-id 1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#split-horizon group
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#vfi abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#neighbor 192.168.99.17 pw-id 1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#show
Mon Oct 18 13:51:05.831 EDT
l2vpn
 bridge group examples
  bridge-domain all_three
   interface GigabitEthernet0/0/0/0.99
   !
   interface GigabitEthernet0/0/0/0.101
    split-horizon group
   !
   neighbor 192.168.99.1 pw-id 1
   !
   neighbor 192.168.99.9 pw-id 1
    split-horizon group
   !
   vfi abc
    neighbor 192.168.99.17 pw-id 1
    !
   !
  !
  !
!

RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

According to this example, the Split Horizon group assignments for bridge domain **all_three** are:

Bridge Port/Pseudowire	Split Horizon Group
bridge port: gig0/0/0/0.99	0
bridge port: gig0/0/0/0.101	2
PW: 192.168.99.1 pw-id 1	0
PW: 192.168.99.9 pw-id 1	2
PW: 192.168.99.17 pw-id 1	1

Blocking Unknown Unicast Flooding: Example

Unknown-unicast flooding can be blocked at these levels:

- bridge domain
- bridge port (attachment circuit (AC))
- access pseudowire (PW)

This example shows how to block unknown-unicast flooding at the bridge domain level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    flooding unknown-unicast disable
  end
```

This example shows how to block unknown-unicast flooding at the bridge port level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    interface GigabitEthernet 0/1/0/1
    flooding unknown-unicast disable
  end
```

This example shows how to block unknown-unicast flooding at the access pseudowire level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    neighbor 10.1.1.1 pw-id 1000
    flooding unknown-unicast disable
  end
```

Disabling MAC Flush: Examples

You can disable the MAC flush at these levels:

- bridge domain
- bridge port (attachment circuit (AC))
- access pseudowire (PW)

The following example shows how to disable the MAC flush at the bridge domain level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    mac
    port-down flush disable
  end
```

The following example shows how to disable the MAC flush at the bridge port level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    interface TenGigE 0/0/0/0
    mac
    port-down flush disable
  end
```

Bridging on IOS XR Trunk Interfaces: Example

This example shows how to configure a as a simple L2 switch.

Important notes:

Create a bridge domain that has four attachment circuits (AC). Each AC is an IOS XR trunk interface (i.e. not a subinterface/EFP).

- This example assumes that the running config is empty, and that all the components are created.
- This example provides all the necessary steps to configure the to perform switching between the interfaces. However, the commands to prepare the interfaces such as no shut, negotiation auto, etc., have been excluded.
- The bridge domain is in a no shut state, immediately after being created.
- Only trunk (i.e. main) interfaces are used in this example.
- The trunk interfaces are capable of handling tagged (i.e. IEEE 802.1Q) or untagged (i.e. no VLAN header) frames.
- The bridge domain learns, floods, and forwards based on MAC address. This functionality works for frames regardless of tag configuration.
- The bridge domain entity spans the entire system. It is not necessary to place all the bridge domain ACs on a single LC. This applies to any bridge domain configuration.

- The show bundle and the show l2vpn bridge-domain commands are used to verify that the router was configured as expected, and that the commands show the status of the new configurations.
- The ACs in this example use interfaces that are in the admin down state.

Configuration Example

```
RP/0/RSP0/CPU0:router#config
RP/0/RSP0/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/5
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/6
RP/0/RSP0/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP0/CPU0:router(config-if)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-if)#l2transport
RP/0/RSP0/CPU0:router(config-if-l2)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain test-switch
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/0
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/2/0/1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP0/CPU0:Jul 26 10:48:21.320 EDT: config[65751]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user 'lab'. Use 'show configuration commit changes 1000000973'
to view the changes.
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP0/CPU0:Jul 26 10:48:21.342 EDT: config[65751]: %MGBL-SYS-5-CONFIG_I : Configured
from console by lab
RP/0/RSP0/CPU0:router#show bundle Bundle-ether10

Bundle-Ether10
  Status:                                     Down
  Local links <active/standby/configured>:    0 / 0 / 2
  Local bandwidth <effective/available>:      0 (0) kbps
  MAC address (source):                       0024.f71e.22eb (Chassis pool)
  Minimum active links / bandwidth:           1 / 1 kbps
  Maximum active links:                       64
  Wait while timer:                           2000 ms
  LACP:                                       Operational
    Flap suppression timer:                   Off
  mLACP:                                      Not configured
  IPv4 BFD:                                   Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------  ----------
  Gi0/2/0/5             Local            Configured   0x8000, 0x0001     1000000
      Link is down
  Gi0/2/0/6             Local            Configured   0x8000, 0x0002     1000000
      Link is down

RP/0/RSP0/CPU0:router#
RP/0/RSP0/CPU0:router#show l2vpn bridge-domain group examples
Bridge group: examples, bridge-domain: test-switch, id: 2000, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 4 (1 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
  List of ACs:
    BE10, state: down, Static MAC addresses: 0
    Gi0/2/0/0, state: up, Static MAC addresses: 0
    Gi0/2/0/1, state: down, Static MAC addresses: 0
    Te0/5/0/1, state: down, Static MAC addresses: 0
  List of VFIs:
RP/0/RSP0/CPU0:router#
```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

SUMMARY STEPS

1. **configure**
2. **interface Bundle-ether10**
3. **l2transport**
4. **interface GigabitEthernet0/2/0/5**
5. **bundle id 10 mode active**
6. **interface GigabitEthernet0/2/0/6**
7. **bundle id 10 mode active**
8. **interface GigabitEthernet0/2/0/0**
9. **l2transport**
10. **interface GigabitEthernet0/2/0/1**
11. **l2transport**
12. **interface TenGigE0/1/0/2**
13. **l2transport**
14. **l2vpn**
15. **bridge group examples**
16. **bridge-domain test-switch**
17. **interface Bundle-ether10**
18. **exit**
19. **interface GigabitEthernet0/2/0/0**
20. **exit**
21. **interface GigabitEthernet0/2/0/1**
22. **exit**
23. **interface TenGigE0/1/0/2**
24. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Enters global configuration mode.

Step 2 **interface Bundle-ether10**

Creates a new bundle trunk interface.

Step 3 **l2transport**

Changes Bundle-ether10 from an L3 interface to an L2 interface.

Step 4 **interface GigabitEthernet0/2/0/5**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/5.

Step 5 **bundle id 10 mode active**

Establishes GigabitEthernet0/2/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 6 **interface GigabitEthernet0/2/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/6.

Step 7 **bundle id 10 mode active**

Establishes GigabitEthernet0/2/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 8 **interface GigabitEthernet0/2/0/0**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/0.

Step 9 **l2transport**

Change GigabitEthernet0/2/0/0 from an L3 interface to an L2 interface.

Step 10 **interface GigabitEthernet0/2/0/1**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/2/0/1.

Step 11 **l2transport**

Change GigabitEthernet0/2/0/1 from an L3 interface to an L2 interface.

Step 12 **interface TenGigE0/1/0/2**

Enters interface configuration mode. Changes configuration mode to act on TenGigE0/1/0/2.

Step 13 **l2transport**

Changes TenGigE0/1/0/2 from an L3 interface to an L2 interface.

Step 14 **l2vpn**

Enters L2VPN configuration mode.

Step 15 **bridge group examples**

Creates the bridge group **examples**.

Step 16 **bridge-domain test-switch**

Creates the bridge domain **test-switch**, that is a member of bridge group **examples**.

Step 17 **interface Bundle-ether10**

Establishes Bundle-ether10 as an AC of bridge domain test-switch.

Step 18 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 19 **interface GigabitEthernet0/2/0/0**

Establishes GigabitEthernet0/2/0/0 as an AC of bridge domain **test-switch**.

Step 20 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 21 **interface GigabitEthernet0/2/0/1**

Establishes GigabitEthernet0/2/0/1 as an AC of bridge domain **test-switch**.

Step 22 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 23 **interface TenGigE0/1/0/2**

Establishes interface TenGigE0/1/0/2 as an AC of bridge domain **test-switch**.

Step 24 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Bridging on Ethernet Flow Points: Example

This example shows how to configure a to perform Layer 2 switching on traffic that passes through Ethernet Flow Points (EFPs). EFP traffic typically has one or more VLAN headers. Although both IOS XR trunks and IOS XR EFPs can be combined as attachment circuits in bridge domains, this example uses EFPs exclusively.

Important notes:

- An EFP is a Layer 2 subinterface. It is always created under a trunk interface. The trunk interface must exist before the EFP is created.
- In an empty configuration, the bundle interface trunk does not exist, but the physical trunk interfaces are automatically configured. Therefore, only the bundle trunk is created.
- In this example the subinterface number and the VLAN IDs are identical, but this is out of convenience, and is not a necessity. They do not need to be the same values.
- The bridge domain test-efp has three attachment circuits (ACs). All the ACs are EFPs.
- Only frames with a VLAN ID of 999 enter the EFPs. This ensures that all the traffic in this bridge domain has the same VLAN encapsulation.
- The ACs in this example use interfaces that are in the admin down state (**unresolved** state). Bridge domains that use nonexistent interfaces as ACs are legal, and the commit for such configurations does not fail. In this case, the status of the bridge domain shows **unresolved** until you configure the missing interface.

Configuration Example

```
RP/0/RSP1/CPU0:router#configure
RP/0/RSP1/CPU0:router(config)#interface Bundle-ether10
RP/0/RSP1/CPU0:router(config-if)#interface Bundle-ether10.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface GigabitEthernet0/6/0/5
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/6
RP/0/RSP1/CPU0:router(config-if)#bundle id 10 mode active
RP/0/RSP1/CPU0:router(config-if)#interface GigabitEthernet0/6/0/7.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#interface TenGigE0/1/0/2.999 l2transport
RP/0/RSP1/CPU0:router(config-subif)#encapsulation dot1q 999
RP/0/RSP1/CPU0:router(config-subif)#l2vpn
RP/0/RSP1/CPU0:router(config-l2vpn)#bridge group examples
RP/0/RSP1/CPU0:router(config-l2vpn-bg)#bridge-domain test-efp
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-ether10.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/6/0/7.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#exit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd)#interface TenGigE0/1/0/2.999
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#commit
RP/0/RSP1/CPU0:router(config-l2vpn-bg-bd-ac)#end
RP/0/RSP1/CPU0:router#
RP/0/RSP1/CPU0:router#show l2vpn bridge group examples
Fri Jul 23 21:56:34.473 UTC Bridge group: examples, bridge-domain: test-efp, id: 0, state:
 up, ShgId: 0, MSTi: 0
Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 3 (0 up), VFIs: 0, PWs: 0 (0 up), PBBs: 0 (0 up)
  List of ACs:
    BE10.999, state: down, Static MAC addresses: 0
    Gi0/6/0/7.999, state: unresolved, Static MAC addresses: 0
    Te0/1/0/2.999, state: down, Static MAC addresses: 0
  List of VFIs:
RP/0/RSP1/CPU0:router#
```

This table lists the configuration steps (actions) and the corresponding purpose for this example:

SUMMARY STEPS

1. **configure**
2. **interface Bundle-ether10**
3. **interface Bundle-ether10.999 l2transport**
4. **encapsulation dot1q 999**
5. **interface GigabitEthernet0/6/0/5**
6. **bundle id 10 mode active**
7. **interface GigabitEthernet0/6/0/6**
8. **bundle id 10 mode active**
9. **interface GigabitEthernet0/6/0/7.999 l2transport**
10. **encapsulation dot1q 999**
11. **interface TenGigE0/1/0/2.999 l2transport**
12. **encapsulation dot1q 999**
13. **l2vpn**
14. **bridge group examples**

15. **bridge-domain test-efp**
16. **interface Bundle-ether10.999**
17. **exit**
18. **interface GigabitEthernet0/6/0/7.999**
19. **exit**
20. **interface TenGigE0/1/0/2.999**
21. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Enters global configuration mode.

Step 2 **interface Bundle-ether10**

Creates a new bundle trunk interface.

Step 3 **interface Bundle-ether10.999 l2transport**

Creates an EFP under the new bundle trunk.

Step 4 **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

Step 5 **interface GigabitEthernet0/6/0/5**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/5.

Step 6 **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/5 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 7 **interface GigabitEthernet0/6/0/6**

Enters interface configuration mode. Changes configuration mode to act on GigabitEthernet0/6/0/6.

Step 8 **bundle id 10 mode active**

Establishes GigabitEthernet0/6/0/6 as a member of Bundle-ether10. The **mode active** keywords specify LACP protocol.

Step 9 **interface GigabitEthernet0/6/0/7.999 l2transport**

Creates an EFP under GigabitEthernet0/6/0/7.

Step 10 **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

Step 11 **interface TenGigE0/1/0/2.999 l2transport**

Creates an EFP under TenGigE0/1/0/2.

Step 12 **encapsulation dot1q 999**

Assigns VLAN ID of 999 to this EFP.

Step 13 **l2vpn**

Enters L2VPN configuration mode.

Step 14 **bridge group examples**

Creates the bridge group named **examples**.

Step 15 **bridge-domain test-efp**

Creates the bridge domain named **test-efp**, that is a member of bridge group **examples**.

Step 16 **interface Bundle-ether10.999**

Establishes Bundle-ether10.999 as an AC of the bridge domain named **test-efp**.

Step 17 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 18 **interface GigabitEthernet0/6/0/7.999**

Establishes GigabitEthernet0/6/0/7.999 as an AC of the bridge domain named **test-efp**.

Step 19 **exit**

Exits bridge domain AC configuration submode, allowing next AC to be configured.

Step 20 **interface TenGigE0/1/0/2.999**

Establishes interface TenGigE0/1/0/2.999 as an AC of bridge domain named **test-efp**.

Step 21 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Changing the Flood Optimization Mode

Perform this task to change the flood optimization mode under the bridge domain:

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **flood mode convergence-optimized**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group csco
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#
```

Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5 **flood mode convergence-optimized**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# flood mode convergence-optimized
```

Changes the default flood optimization mode from Bandwidth Optimization Mode to Convergence Mode.

When you configure **flood mode convergence-optimized**, you must remove and reconfigure the bridge domain when you add, modify, or remove the pseudowire configuration of a specific bridge domain.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

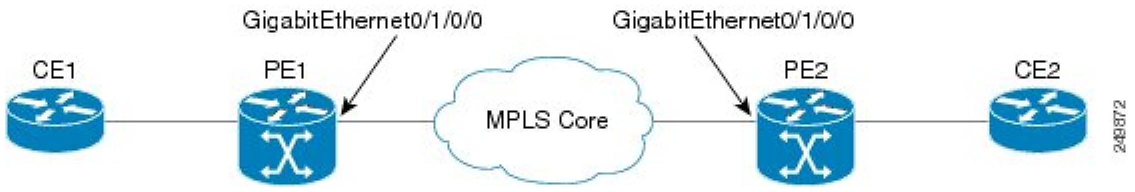
Configuring VPLS with BGP Autodiscovery and Signaling: Example

This section contains these configuration examples for configuring the BGP autodiscovery and signaling feature:

LDP and BGP Configuration

The following figure illustrates an example of LDP and BGP configuration.

Figure 37: LDP and BGP Configuration



Configuration at PE1:

```
interface Loopback0
 ipv4 address 10.0.0.100 255.0.0.0
!
interface Loopback1
 ipv4 address 10.0.0.1 255.0.0.0
!
mpls ldp
 router-id 10.0.0.10
 interface GigabitEthernt0/1/0/0
!
router bgp 120
 address-family l2vpn vpls-vpws
!
 neighbor 172.16.0.1
 remote-as 120
 update-source Loopback1
 address-family l2vpn vpls-vpws
 signaling bgp disable
```

Configuration at PE2:

```
interface Loopback0
 ipv4 address 172.16.0.10 255.255.255.255
!
interface Loopback1
 ipv4 address 172.16.0.1 255.255.255.255
!
mpls ldp
 router-id 172.16.0.100
 interface GigabitEthernt0/1/0/0
!
router bgp 120
```

```
 address-family l2vpn vpls-vpws
!
 neighbor 10.0.0.11
 remote-as 120
 update-source Loopback1
 address-family l2vpn vpls-vpws
```

Minimum L2VPN Configuration for BGP Autodiscovery with BGP Signaling

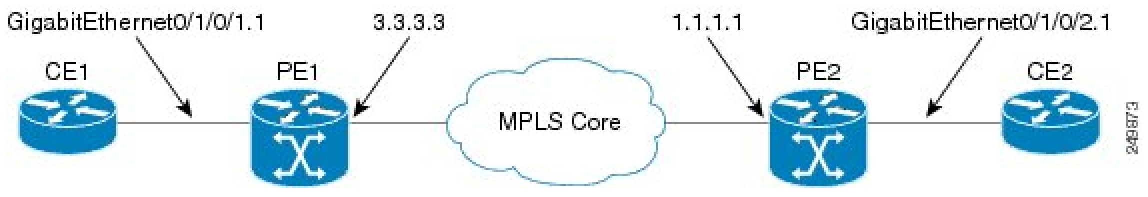
This example illustrates the minimum L2VPN configuration required for BGP Autodiscovery with BGP Signaling, where any parameter that has a default value is not configured.

```
(config)# l2vpn
(config-l2vpn)# bridge group {bridge group name}
(config-l2vpn-bg)# bridge-domain {bridge domain name}
(config-l2vpn-bg-bd)# vfi {vfi name}
(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
(config-l2vpn-bg-bd-vfi-ad)# vpn-id 10
(config-l2vpn-bg-bd-vfi-ad)# rd auto
(config-l2vpn-bg-bd-vfi-ad)# route-target 10.0.0.1:100
(config-l2vpn-bg-bd-vfi-ad-sig)# signaling-protocol bgp
(config-l2vpn-bg-bd-vfi-ad-sig)# ve-id 1
(config-l2vpn-bg-bd-vfi-ad-sig)# commit
```

VPLS with BGP Autodiscovery and BGP Signaling

The following figure illustrates an example of configuring VPLS with BGP autodiscovery (AD) and BGP Signaling.

Figure 38: VPLS with BGP autodiscovery and BGP signaling



Configuration at PE1:

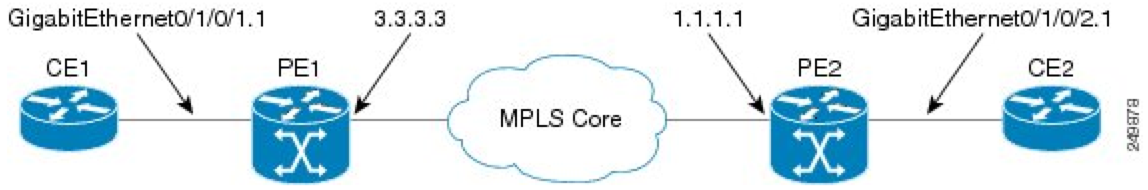
```
l2vpn
  bridge group gr1
   bridge-domain bd1
     interface GigabitEthernet0/1/0/1.1
     vfi vf1
     ! AD independent VFI attributes
     vpn-id 100
     ! Auto-discovery attributes
     autodiscovery bgp
     rd auto
     route-target 2.2.2.2:100
     ! Signaling attributes
     signaling-protocol bgp
     ve-id 3
```

Configuration at PE2:

```
l2vpn
```

```
 bridge group gr1
  bridge-domain bd1
   interface GigabitEthernet0/1/0/2.1
   vfi vf1
   ! AD independent VFI attributes
   vpn-id 100
   ! Auto-discovery attributes
   autodiscovery bgp
   rd auto
   route-target 2.2.2.2:100
   ! Signaling attributes
   signaling-protocol bgp
   ve-id 5
```

This is an example of NLRI for VPLS with BGP AD and signaling:



Discovery Attributes

NLRI sent at PE1:

```
Length = 19
Router Distinguisher = 3.3.3.3:32770
VE ID = 3
VE Block Offset = 1
VE Block Size = 10
Label Base = 16015
```

NLRI sent at PE2:

```
Length = 19
Router Distinguisher = 1.1.1.1:32775
VE ID = 5
VE Block Offset = 1
VE Block Size = 10
Label Base = 16120
```

Minimum Configuration for BGP Autodiscovery with LDP Signaling

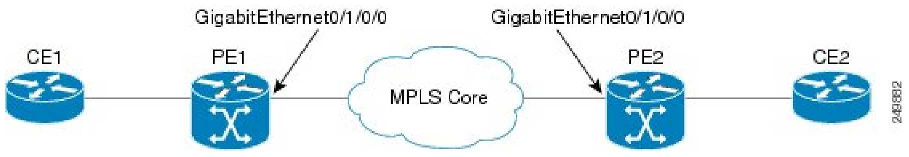
This example illustrates the minimum L2VPN configuration required for BGP Autodiscovery with LDP Signaling, where any parameter that has a default value is not configured.

```
(config)# l2vpn
(config-l2vpn)# bridge group {bridge group name}
(config-l2vpn-bg)# bridge-domain {bridge domain name}
(config-l2vpn-bg-bd)# vfi {vfi name}
(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
(config-l2vpn-bg-bd-vfi-ad)# vpn-id 10
(config-l2vpn-bg-bd-vfi-ad)# rd auto
(config-l2vpn-bg-bd-vfi-ad)# route-target 1.1.1.1:100
(config-l2vpn-bg-bd-vfi-ad)# commit
```

VPLS with BGP Autodiscovery and LDP Signaling

The following figure illustrates an example of configuring VPLS with BGP autodiscovery (AD) and LDP Signaling.

Figure 39: VPLS with BGP autodiscovery and LDP signaling



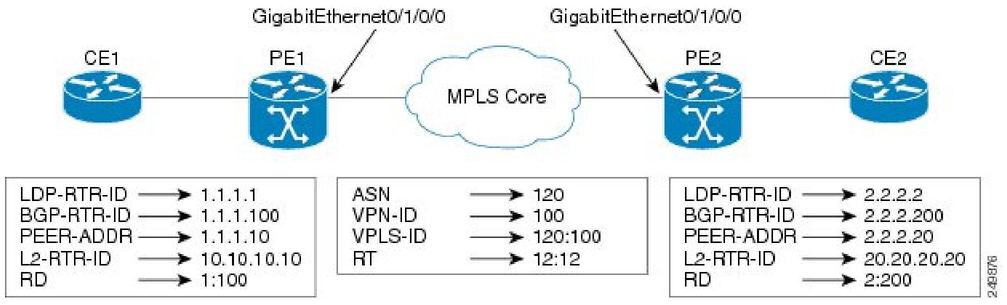
Configuration at PE1:

```
l2vpn
  router-id 10.10.10.10
  bridge group bg1
   bridge-domain bd1
    vfi vf1
    vpn-id 100
    autodiscovery bgp
    rd 1:100
    router-target 12:12
```

Configuration at PE2:

```
l2vpn
  router-id 20.20.20.20
  bridge group bg1
   bridge-domain bd1
    vfi vf1
     vpn-id 100
     autodiscovery bgp
      rd 2:200
      router-target 12:12
      signaling-protocol ldp
       vpls-id 120:100
```

Discovery and Signaling Attributes



Configuration at PE1:

```
LDP Router ID - 10.0.0.1
BGP Router ID - 10.0.0.100
```

```
Peer Address -  10.0.0.10
L2VPN Router ID - 10.10.10.10
Route Distinguisher - 1:100
```

Common Configuration between PE1 and PE2:

```
ASN - 120
VPN ID - 100
VPLS ID - 120:100
Route Target - 12:12
```

Configuration at PE2:

```
LDP Router ID - 172.16.0.1
BGP Router ID - 172.16.0.100
Peer Address -  172.16.0.10
L2VPN Router ID - 192.168.0.1
Route Distinguisher - 2:200
```

Discovery Attributes

NLRI sent at PE1:

```
Source Address - 10.0.0.10
Destination Address - 172.16.0.10
Length - 14
Route Distinguisher - 1:100
L2VPN Router ID - 10.10.10.10
VPLS ID - 120:100
Route Target - 12:12
```

NLRI sent at PE2:

```
Source Address - 172.16.0.10
Destination Address - 10.0.0.10
Length - 14
Route Distinguisher - 2:200
L2VPN Router ID - 192.168.0.1
VPLS ID - 120:100
Route Target - 12:12
```

Enabling VC type 4 for BGP Autodiscovery

This example shows how to configure virtual connection type 4 in VPLS with BGP autodiscovery:

```
l2vpn
bridge group bg1
  bridge-domain bd1
   transport-mode vlan passthrough
   interface GigabitEthernet0/0/0/1.1
   !
   neighbor 172.16.0.1 pw-id 1
   !
   vfi vf1
    vpn-id 100
    autodiscovery bgp
```

```
     rd auto
     route-target 1:1
     signalining-protocol ldp
    !
   !
  !
!
```

VPLS with both BGP Autodiscovery and Manual Provisioning of VPLS Peers

This example shows VPLS configuration with BGP autodiscovery as well as manual provisioning of VPLS peers that are not participating in the BGP autodiscovery process.

```
!
l2vpn
 bridge group bg1
  bridge-domain bd1
   !
   vfi vfi1
    vpn-id 500
    autodiscovery bgp
     rd auto
     route-target 65533:12345678
     signaling-protocol ldp
      vpls-id 65533:12345678
    ! Manually provisioned peers
    neighbor 10.10.10.2 pw-id 102
!
```

Configuring Dynamic ARP Inspection: Example

This example shows how to configure basic dynamic ARP inspection under a bridge domain:

```
config
l2vpn
 bridge group MyGroup
 bridge-domain MyDomain
 dynamic-arp-inspection logging
```

This example shows how to configure basic dynamic ARP inspection under a bridge port:

```
config
l2vpn
 bridge group MyGroup
 bridge-domain MyDomain
  interface gigabitEthernet 0/1/0/0.1
 dynamic-arp-inspection logging
```

This example shows how to configure optional dynamic ARP inspection under a bridge domain:

```
l2vpn
  bridge group SECURE
    bridge-domain SECURE-DAI
      dynamic-arp-inspection
        logging
        address-validation
          src-mac
          dst-mac
           ipv4
```

This example shows how to configure optional dynamic ARP inspection under a bridge port:

```
l2vpn
  bridge group SECURE
```

```
    bridge-domain SECURE-DAI
      interface GigabitEthernet0/0/0/1.10
        dynamic-arp-inspection
          logging
          address-validation
           src-mac
           dst-mac
           ipv4
```

This example shows the output of the **show l2vpn bridge-domain** *bd-name* SECURE-DAI **detail** command:

```
#show l2vpn bridge-domain bd-name SECURE-DAI detail
Bridge group: SECURE, bridge-domain: SECURE-DAI, id: 2, state: up,
…
Dynamic ARP Inspection: enabled, Logging: enabled
Dynamic ARP Inspection Address Validation:
   IPv4 verification: enabled
   Source MAC verification: enabled
    Destination MAC verification: enabled
…
List of ACs:
 AC: GigabitEthernet0/0/0/1.10, state is up
…
    Dynamic ARP Inspection: enabled, Logging: enabled
    Dynamic ARP Inspection Address Validation:
        IPv4 verification: enabled
        Source MAC verification: enabled
        Destination MAC verification: enabled
        IP Source Guard: enabled, Logging: enabled

…
   Dynamic ARP inspection drop counters:
      packets: 1000, bytes: 64000
```

This example shows the output of the **show l2vpn forwarding interface** *interface-name* **detail location** *location-name* command:

```
#show l2vpn forwarding interface g0/0/0/1.10 det location 0/0/CPU0
Local interface: GigabitEthernet0/0/0/1.10, Xconnect id: 0x40001, Status: up

…
    Dynamic ARP Inspection: enabled, Logging: enabled
    Dynamic ARP Inspection Address Validation:
        IPv4 verification: enabled
        Source MAC verification: enabled
        Destination MAC verification: enabled
        IP Source Guard: enabled, Logging: enabled
```

This example shows the logging display:

```
LC/0/0/CPU0:Jun 16 13:28:28.697 : l2fib[188]: %L2-L2FIB-5-SECURITY_DAI_VIOLATION_AC : Dynamic
 ARP inspection in AC GigabitEthernet0_0_0_7.1000 detected violated packet - source MAC:
0000.0000.0065, destination MAC: 0000.0040.0000, sender MAC: 0000.0000.0064, target MAC:
0000.0000.0000, sender IP: 5.6.6.6, target IP: 130.10.3.2

LC/0/5/CPU0:Jun 16 13:28:38.716 : l2fib[188]: %L2-L2FIB-5-SECURITY_DAI_VIOLATION_AC : Dynamic
 ARP inspection in AC Bundle-Ether100.103 detected violated packet - source MAC:
0000.0000.0067, destination MAC: 0000.2300.0000, sender MAC: 0000.7800.0034, target MAC:
0000.0000.0000, sender IP: 130.2.5.1, target IP: 50.5.1.25
```

Configuring IP Source Guard: Example

This example shows how to configure basic IP source guard under a bridge domain:

```
config
l2vpn
 bridge group MyGroup
 bridge-domain MyDomain
 ip-source-guard logging
```

This example shows how to configure basic IP source guard under a bridge port:

```
config
l2vpn
 bridge group MyGroup
 bridge-domain MyDomain
  interface gigabitEthernet 0/1/0/0.1
 ip-source-guard  logging
```

This example shows how to configure optional IP source guard under a bridge domain:

```
l2vpn
  bridge group SECURE
    bridge-domain SECURE-IPSG
       ip-source-guard
         logging
```

This example shows how to configure optional IP source guard under a bridge port:

```
l2vpn
  bridge group SECURE
    bridge-domain SECURE-IPSG
      interface GigabitEthernet0/0/0/1.10
        ip-source-guard
          logging
```

This example shows the output of the **show l2vpn bridge-domain** *bd-name ipsg-name* **detail** command:

```
# show l2vpn bridge-domain bd-name SECURE-IPSG detail
Bridge group: SECURE, bridge-domain: SECURE-IPSG, id: 2, state: up,
…
  IP Source Guard: enabled, Logging: enabled
…
List of ACs:
 AC: GigabitEthernet0/0/0/1.10, state is up
…

   IP Source Guard: enabled, Logging: enabled
…
   IP source guard drop counters:
      packets: 1000, bytes: 64000
```

This example shows the output of the **show l2vpn forwarding interface** *interface-name* **detail location** *location-name* command:

```
# show l2vpn forwarding interface g0/0/0/1.10 detail location 0/0/CPU0
Local interface: GigabitEthernet0/0/0/1.10, Xconnect id: 0x40001, Status: up

…
    IP Source Guard: enabled, Logging: enabled
```

This example shows the logging display:

```
LC/0/0/CPU0:Jun 16 13:32:25.334 : l2fib[188]: %L2-L2FIB-5-SECURITY_IPSG_VIOLATION_AC : IP
```

```
source guard in AC  GigabitEthernet0_0_0_7.1001 detected violated packet - source MAC:
0000.0000.0200, destination MAC: 0000.0003.0000, source IP: 130.0.0.1, destination IP:
125.34.2.5



LC/0/5/CPU0:Jun 16 13:33:25.530 : l2fib[188]: %L2-L2FIB-5-SECURITY_IPSG_VIOLATION_AC : IP
source guard in AC Bundle-Ether100.100  detected violated packet - source MAC: 0000.0000.0064,
 destination MAC: 0000.0040.0000, source IP: 14.5.1.3, destination IP: 45.1.1.10
```

Configuring G.8032 Ethernet Ring Protection: Example

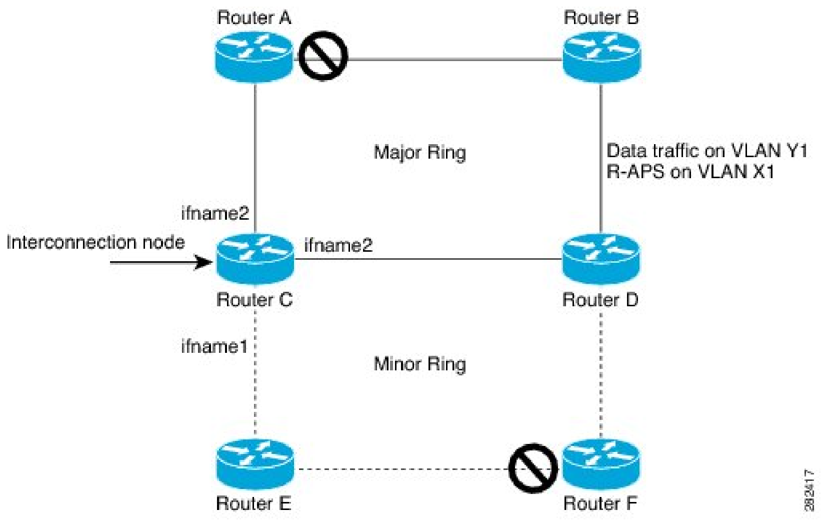
This sample configuration illustrates the elements that a complete G.8032 configuration includes:

```
# Configure the ERP profile characteristics if ERP instance behaviors are non-default.
ethernet ring g8032 profile ERP-profile
  timer wtr 60
  timer guard 100
  timer hold-off 1
  non-revertive

# Configure CFM MEPs and configure to monitor the ring links.
ethernet cfm
  domain domain1
    service link1 down-meps
   continuity-check interval 100ms
      efd
   mep crosscheck
  mep-id 2
  domain domain2
    service link2 down-meps
   continuity-check interval 100ms
      efd protection-switching
   mep crosscheck
  mep id 2

Interface Gig 0/0/0/0
  ethernet cfm mep domain domain1 service link1 mep-id 1
Interface Gig 1/1/0/0
  ethernet cfm mep domain domain2 service link2 mep-id 1

# Configure the ERP 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 40: Open Ring Scenario - interconnection node



The minimum configuration required for configuring G.8032 at Router C (Open ring – Router C):

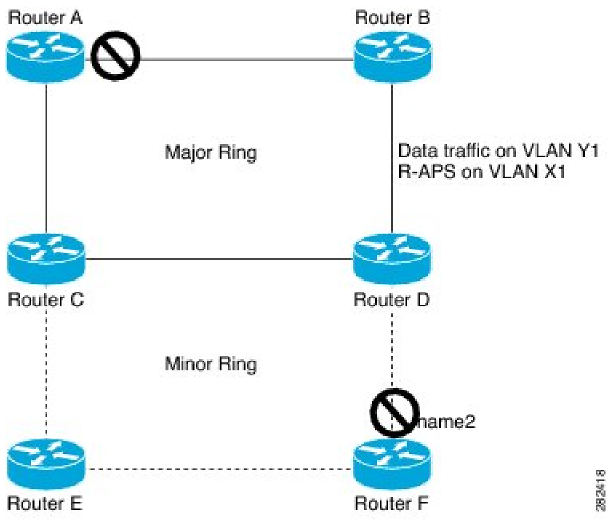
```
interface <ifname1.1> l2transport
 encapsulation dot1q X1
interface <ifname1.10> l2transport
 encapsulation dot1q Y1
interface <ifname2.10> l2transport
 encapsulation dot1q Y1
interface <ifname3.10> l2transport
 encapsulation dot1q Y1
l2vpn
ethernet ring g8032 <ring-name>
      port0 interface <main 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 41: 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>
```

Configuring Flow Aware Transport Pseudowire: Example

This sample configuration shows how to enable load balancing with FAT PW for VPWS.

```
l2vpn
pw-class class1
    encapsulation mpls
        load-balancing flow-label transmit
    !
 !
pw-class class2
    encapsulation mpls
        load-balancing flow-label both
    !

xconnect group group1
    p2p p1
        interface TenGigE 0/0/0/0.1
        neighbor 1.1.1.1 pw-id 1
            pw-class class1
        !
    !
!
```

This sample configuration shows how to enable load balancing with FAT PW for VPLS.

Note For VPLS, the configuration at the bridge-domain level is applied to all PWs (access and VFI PWs). Pseudowire classes are defined to override the configuration for manual PWs.

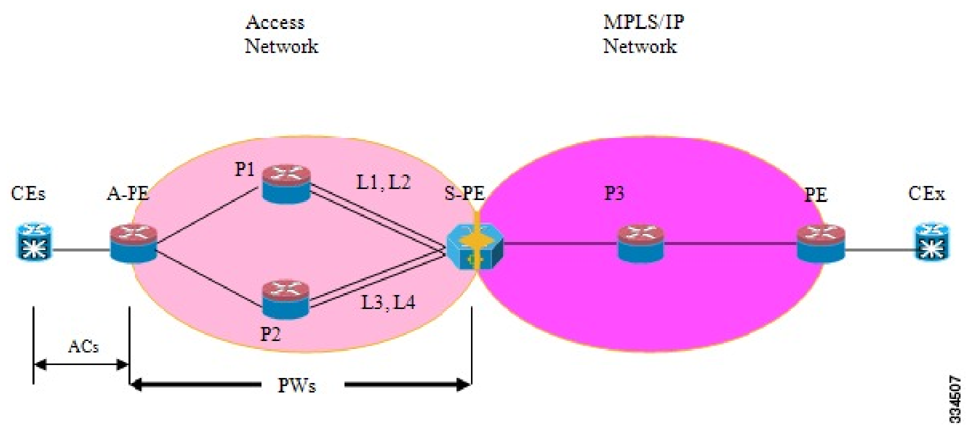
```
l2vpn
pw-class class1
 encapsulation mpls
 load-balancing flow-label both

bridge group group1
    bridge-domain domain1
        vfi vfi2-auto-bgp
                autodiscovery bgp
                   signaling-protocol bgp
                       load-balancing flow-label both static
               !
            !
        !
    !
    bridge-domain domain2
        vfi vfi2-auto-ldp
                autodiscovery bgp
                   signaling-protocol ldp
                       load-balancing flow-label both static
               !
            !
        !
    !
!
```

Configuring Pseudowire Headend: Example

This example shows how to configure pseudowire headend.

Consider the topology in the following figure.

Figure 42: Pseudowire Headend Example



There are multiple CEs connected to A-PE (each CE is connected by one link). There are two P routers between A-PE an S-PE in the access network. The S-PE is connected using two links to P1. These links L1 and L2 (on two different line cards on P1 and S-PE), e.g. Gig0/1/0/0 and Gig0/2/0/0 respectively.

The S-PE is connected by two links to P2, links L3 and L4 (on two different line cards on P2 and S-PE), e.g. Gig0/1/0/1 and Gig0/2/0/1 respectively. For each CE-APE link, a xconnect (AC-PW) is configured on the A-PE. A-PE uses router-id 100.100.100.100 for routing and PW signaling. Two router-ids on S-PE used for PW signaling, e.g. 111.111.111.111 and 112.112.112.112 (for rx pin-down), 110.110.110.110 is the router-id for routing.

CE Configuration

Consider two CEs that are connected through Ge0/3/0/0 (CE1 and A-PE) and Ge0/3/0/1 (CE2 and A-PE).

CE1

```
interface Gig0/3/0/0
 ipv4 address 10.0.0.1/8
router static
 address-family ipv4 unicast
  110.110.110.110 Gig0/3/0/0
  A.B.C.D/N 110.110.110.110
```

CE2

```
interface Gig0/3/0/1
 ipv4 address 10.1.2.1/24
router static
 address-family ipv4 unicast
```

```
  110.110.110.110 Gig0/3/0/1
  A.B.C.D/N 110.110.110.110
```

A-PE Configuration

At A-PE we have 1 xconnect for each CE connection. Here we give the configuration for the 2 CE connections above: both connections are L2 links which are in xconnects. Each xconnect has a PW destined to S-PE but we use a different neighbor address depending of where we want to pin-down the PW: [L1, L4] or [L2, L3]

```
interface Gig0/3/0/0
 l2transport
interface Gig0/3/0/1
 l2transport

l2vpn
 xconnect group pwhe
  p2p pwhe_spe_1
   interface Gig0/3/0/0
   neighbor 111.111.111.111 pw-id 1
  p2p pwhe_spe_2
   interface Gig0/3/0/1
   neighbor 112.112.112.112 pw-id 2
```

P Router Configuration

We need static routes on P routers for rx pindown on S-PE, i.e. to force PWs destined to a specific address to be transported over certain links.

P1

```
router static
 address-family ipv4 unicast
  111.111.111.111 Gig0/1/0/0
  112.112.112.112 Gig0/2/0/0
```

P2

```
router static
 address-family ipv4 unicast
  111.111.111.111 Gig0/2/0/1
  112.112.112.112 Gig0/1/0/1
```

S-PE Configuration

At S-PE we have 2 PW-HE interfaces (1 for each PW) and each uses a different interface list for tx pin-down (has to match the static config at P routers for rx pin-down). Each PW-HE has its PW going to A-PE (pw-id has to match what's at A-PE).

```
generic-interface-list il1
 interface gig0/1/0/0
 interface gig0/2/0/0
generic-interface-list il2
 interface gig0/1/0/1
 interface gig0/2/0/1
interface pw-ether1
 ipv4 address 10.1.1.2/24
 attach generic-interface-list il1
interface pw-ether2
 ipv4 address 10.1.2.2/24
 attach generic-interface-list il2

l2vpn
 xconnect group pwhe
  p2p pwhe1
   interface pw-ether1
   neighbor 100.100.100.100 pw-id 1
  p2p pwhe2
   interface pw-ether2
   neighbor 100.100.100.100 pw-id 2
```

Configuring L2VPN over GRE: Example

Configure the PW core interfaces under IGP and ensure that the loopback is reachable. Configure the tunnel source such that the tunnel is the current loopback as well as destination of the peer PE loopback. Now, configure the GRE tunnel in IGP (OSPF or ISIS), and also under **mpls ldp** and ensure that the LDP neighbor is established between the PEs for the PW. This ensures that the PW is Up through the tunnel.

Configuration on PE1:

```
router ospf 1
router-id 1.1.1.1
 area 0
  interface Loopback0
  interface TenGigE0/0/0/1
router ospf 2
 router-id 200.200.200.200
 area 0
  interface Loopback1000
   interface tunnel-ip1
mpls ldp
 router-id 200.200.200.200
 interface tunnel-ip1
```

Configuration on PE2:

```
router ospf 1
 router-id 3.3.3.3
 area 0
  interface Loopback0
interface TenGigE0/2/0/3
router ospf 2
 router-id 201.201.201.201
 area 0
  interface Loopback1000
   interface tunnel-ip1
!
mpls ldp
 router-id 201.201.201.201
 interface tunnel-ip1
```

Configuring a GRE Tunnel as the Preferred Path for Pseudowire: Example

This example shows how to configure a GRE tunnel as the preferred path for a pseudowire.

```
l2vpn
 pw-class gre
```

```
encapsulation mpls
preferred-path interface tunnel-ip 1 fallback disable
```

Configuring a GRE Tunnel as Preferred Path for Pseudowire

Perform this task to configure a GRE tunnel as the preferred path for pseudowires.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** { *name* }
4. **encapsulation mpls**
5. **preferred-path** { **interface** } { **tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value* } [**fallback disable**]
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **pw-class** { *name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pw-class gre
```

Configures the pseudowire class name.

Step 4 **encapsulation mpls**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls
```

Configures the pseudowire encapsulation to MPLS.

Step 5 **preferred-path** { **interface** } { **tunnel-ip** *value* | **tunnel-te** *value* | **tunnel-tp** *value* } [**fallback disable**]

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pwc-encap-
mpls)# preferred-path interface tunnel-ip 1 fallback disable
```

Configures preferred path tunnel settings. If the fallback disable configuration is used and once the TE/TP tunnel is configured as the preferred path goes down, the corresponding pseudowire can also go down.

Note Ensure that fallback is supported.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

CHAPTER 7

Implementing IEEE 802.1ah Provider Backbone Bridge

This module provides conceptual and configuration information for IEEE 802.1ah Provider Backbone Bridge on Cisco ASR 9000 Series Routers. The IEEE 802.1ah standard (Ref [4]) provides a means for interconnecting multiple provider bridged networks to build a large scale end-to-end Layer 2 provider bridged network.

The Cisco ASR 9000 Series Aggregation Services Routers now supports a scenario when the provider backbone bridge is a VPLS network. You can now configure pseudowires in the PBB edge bridge domain and core bridge domain. In either type of bridge domain, the pseudowire functionality remains the same as in the native bridge domain.

Feature History for Implementing IEEE 802.1ah Provider Backbone Bridge

Release	Modification
Release 3.9.1	This feature was introduced on Cisco ASR 9000 Series Routers
Release 4.3.0	Support was added for these features: • Provider Backbone Bridge VPLS • Multiple I-SID Registration Protocol Lite (MIRP Lite)
Release 4.3.2	Support was added for PBB-EVPN feature.
Release 5.1.2	Support was added for MMRP for PBB VPLS Flood Optimization feature.
Release 6.3.2	Support for PBB-EVPN on nV Satellite access interface bundle over ICL (Inter Chassis Link) bundle.

Note PBB EVPN is supported only on Cisco ASR 9000 2nd generation Ethernet line cards and Cisco ASR 9000 3rd generation Ethernet line cards.

PBB EVPN is not supported on Cisco ASR 9000 4th generation Ethernet line cards and Cisco ASR 9000 5th generation Ethernet line cards.

Prerequisites for Implementing 802.1ah Provider Backbone Bridge

This prerequisite applies to implementing 802.1ah Provider Backbone Bridge:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

  If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- You must be familiar with the multipoint bridging concepts. Refer to the Implementing Multipoint Layer 2 Services module.

Information About Implementing 802.1ah Provider Backbone Bridge

To implement 802.1ah, you must understand these concepts:

Benefits of IEEE 802.1ah standard

The benefits of IEEE 802.1ah provider backbone bridges are as follows:

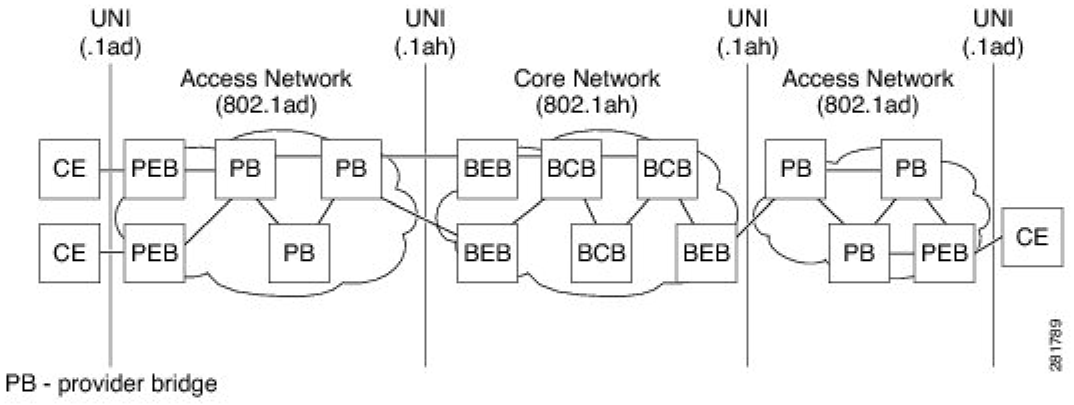
- Increased service instance scalability—Enables a service provider to scale the number of services (service VLANs or service instances) in a Provider Bridged Network (PBN).
- MAC address scalability—Encapsulates the customer packet, including MAC addresses, into a new ethernet frame with new MAC addresses (the backbone bridge MAC addresses). This eliminates the need for backbone core bridges to learn all MAC addresses of every customer and also eases the load on backbone edge bridges.
- VPLS pseudowire reduction and mesh scalability—The number of pseudowires in an IP/MPLS core can be significantly reduced. This is because a single VPLS service can now transport several customer service instances thereby allowing a fewer number of pseudowires in the IP/MPLS core to transport a large number of customer services.
- Layer 2 backbone traffic engineering—Enables explicit controls for Layer 2 traffic engineering by separating service discrimination function and moving it to the I-tags thereby leaving the backbone VLAN to be available for Layer 2 traffic engineering functions.
- Point-to-point service scalability and optimization-enables point-to-point service implementation that includes multiple options for service multiplexing as well as end point discovery.

- Backbone flood traffic reduction—Since there are fewer MAC addresses in the core of the network, the amount of flood traffic in the core network is reduced as there are fewer MAC addresses to be relearnt when MAC tables get flushed due to topology changes.

IEEE 802.1ah Standard for Provider Backbone Bridging Overview

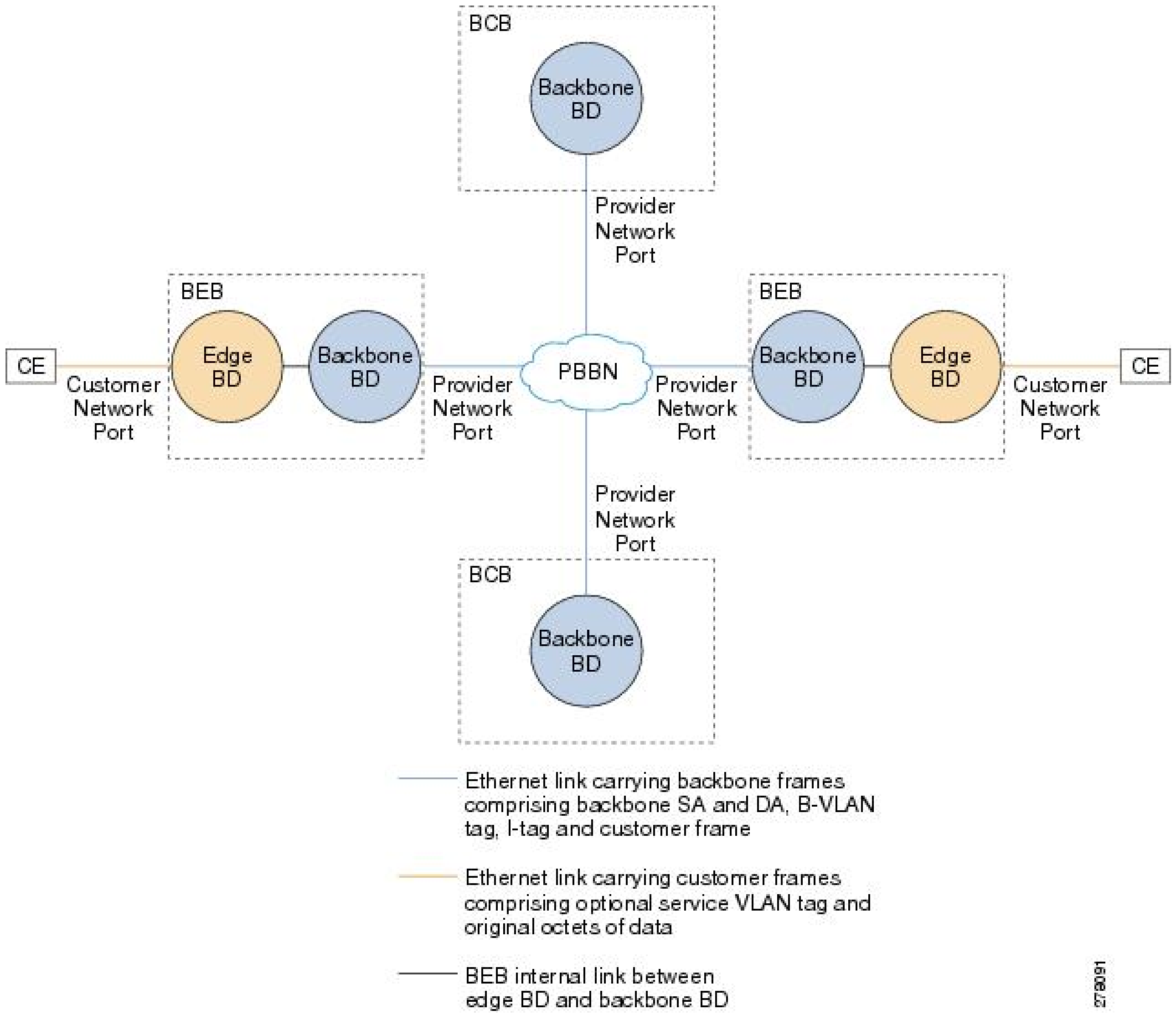
The IEEE 802.1ah Provider Backbone Bridge feature encapsulates or decapsulates end-user traffic on a Backbone Edge Bridge (BEB) at the edge of the Provider Backbone Bridged Network (PBBN). A Backbone Core Bridge (BCB) based network provides internal transport of the IEEE 802.1ah encapsulated frames within the PBBN. The following figure shows a typical 802.1ah PBB network.

Figure 43: IEEE 802.1ah Provider Backbone Bridge



The following figure shows a typical provider backbone network topology.

Figure 44: Provider Back Bone Network Topology


BCB
Backbone BD
Provider Network Port
BEB
Edge BD
Backbone BD
CE
Customer Network Port
Provider Network Port
PBBN
Provider Network Port
BEB
Backbone BD
Edge BD
CE
Customer Network Port
Provider Network Port
BCB
Backbone BD
Ethernet link carrying backbone frames comprising backbone SA and DA, B-VLAN tag, I-tag and customer frame
Ethernet link carrying customer frames comprising optional service VLAN tag and original octets of data
BEB internal link between edge BD and backbone BD
278091

Backbone Edge Bridges

Backbone edge bridges (BEBs) can contain either an I-Component or a B-Component. The I-Component maps service VLAN identifiers (S-VIDs) to service instance identifiers (I-SIDs) and adds a provider backbone bridge (PBB) header without a backbone VLAN tag (B-Tag). The B-Component maps I-SIDs to backbone VIDs (B-VIDs) and adds a PBB header with a B-Tag.

The IEEE 802.1ah standard specifies these three types of BEBs:

- The B-BEB contains the B-Component of the MAC-in-MAC bridge. It validates the I-SIDs and maps the frames onto the backbone VLAN (B-VLAN). It also switches traffic based on the B-VLANS within the core bridge.

- The I-BEB contains the I-Component of the MAC-in-MAC bridge. It performs B-MAC encapsulation and inserts the I-SIDs based on the provider VLAN tags (S-tags), customer VLAN tags (C-tags), or S-tag/C-tag pairs.
- The IB-BEB contains one or more I-Components and a single B-Component interconnected through a LAN segment.

Note Only IB-BEBs are supported on Cisco ASR 9000 Series Routers. Cisco IOS XR supports IB-BEB bridge type at the Edge node.

IB-BEB

The IB-BEB contains both the I-Component and the B-Component. The bridge selects the B-MAC and inserts the I-SID based on the provider VLAN tag (S-tag), the customer VLAN tag (C-tag), or both the S-tag and the C-tag. It validates the I-SIDs and it transmits and receives frames on the B-VLAN.

The IEEE 802.1ah on Provider Backbone Bridges feature supports all services mandated by the IEEE 802.1ah standard and extends the services to provides these additional functionalities:

- S-Tagged Service:
  - In multiplexed environments each S-tag maps to an I-SID and may be retained or removed.
  - In bundled environments multiple S-tags map to the same I-SID and the S-tags must be retained.
- C-Tagged Service:
  - In multiplexed environments each C-tag maps to an I-SID and may be retained or removed.
  - In bundled environments multiple C-tags map to the same I-SID and the C-tags must be retained.
- S/C-Tagged Service:
  - In multiplexed environments each S-tag/C-tag pair maps to an I-SID. The S-tag or the S-tag/C-tag pair may be retained or removed.
  - In bundled environments multiple S-tag/C-tags pairs map to the same I-SID and the S-tag/C-tag pair must be retained.
- Port-based Service
  - A port-based service interface is delivered on a Customer Network Port (CNP). A port-based service interface may attach to a C-VLAN Bridge, 802.1d bridge, router or end-station. The service provided by this interface forwards all frames without an S-Tag over the backbone on a single backbone service instance. A port-based interface discards all frames with an S-Tag that have non-null VLAN IDs.

    This example shows how to configure a port-based service:

    ```
     interface GigabitEthernet0/0/0/10.100 l2transport
     encapsulation untagged
    ```

    --> Creates an EFP for untagged frames.

    ```
    interface GigabitEthernet0/0/0/10.101 l2transport
    encapsulation dot1ad priority-tagged
    ```

--> Creates an EFP for null S-tagged frames.

```
interface GigabitEthernet0/0/0/10.102 l2transport
encapsulation dot1q priority-tagged
```

--> Creates an EFP for null C-tagged frames:

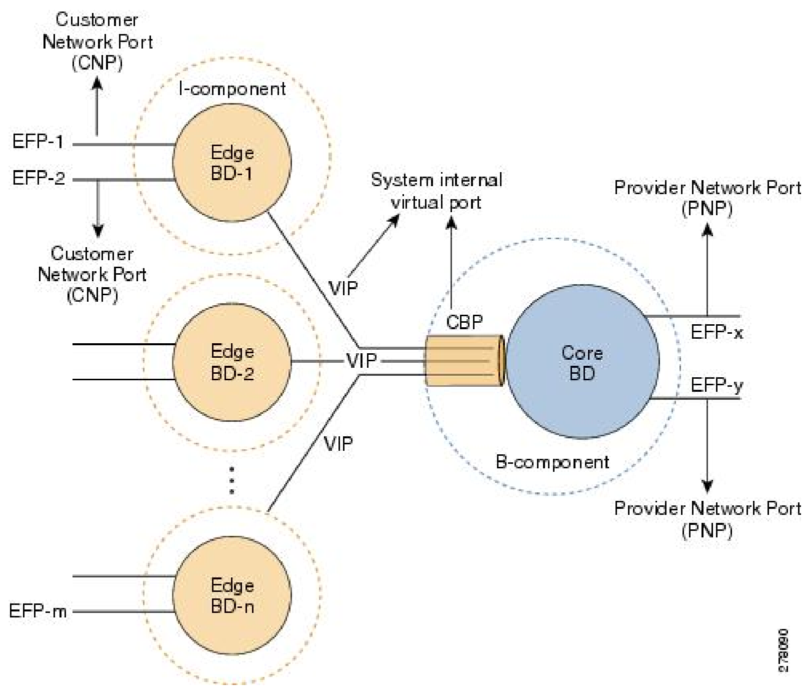
```
interface GigabitEthernet0/0/0/10.103 l2transport
encapsulation dot1q any
```

--> Creates an EFP for C-tagged frames:

Note To configure a port-based service, all the above EFPs must be added to the same edge bridge domain.

The following figure shows the PBB bridge component topology on the Cisco ASR 9000 Series Routers.

Figure 45: PBB Bridge Component Topology on Cisco ASR 9000 Series Routers


Customer Network Port (CNP)
I-component
EFP-1
EFP-2
Edge BD-1
System internal virtual port
Provider Network Port (PNP)
Customer Network Port (CNP)
VIP
CBP
Edge BD-2
VIP
Core BD
EFP-x
EFP-y
VIP
B-component
Provider Network Port (PNP)
Edge BD-n
EFP-m
278090

Multiple I-SID Registration Protocol Lite

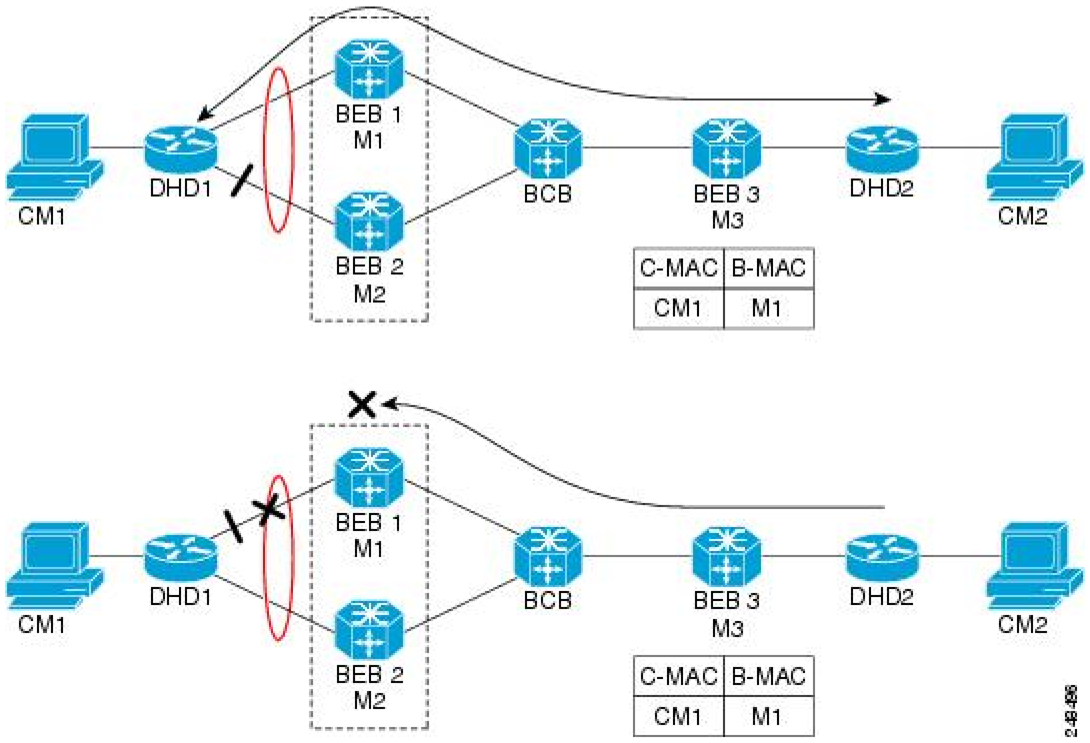
The 802.1Qbe—Multiple I-SID Registration Protocol (MIRP) standard provides the ability to flush learned MAC address registration entries held in the filtering database of an I-component on a per I-SID basis. The backbone service instance identifier (I-SID) is a field in the backbone service instance tag which identifies the backbone service instance of a frame. MIRP defines mechanisms for I-SID flushing, and has the required capabilities to handle topology changes that occur in networks attached to a provider backbone bridged network. A backbone edge bridge (BEB) signals to other potentially affected BEBs, the need to alter certain learned associations between customer MAC addresses and backbone MAC addresses. In the absence of MIRP, customer connections across a provider backbone network can take several minutes to restore connectivity after a topology change in an access network.

In prior releases, PBB traffic was dropped for a MAC aging cycle when bridge forwarding topology changes occurred (due to unavailable ports or spanning tree topology changes) in a PBB edge bridge domain. This resulted in severe limitations for the use of PBB bridges.

Cisco ASR 9000 Series Aggregation Services Routers now support a simplified implementation of the MIRP protocol known as the Multiple I-SID Registration Protocol Lite (MIRP-Lite). The MIRP-Lite feature enables detection of a topology change at a site. A specially defined packet is flooded to all remote edge sites of the PBB network when a site detects a topology change. At the sender site, I-SID of the I-component is placed in the I-TAG of the frame header to specify the I-SID that needs a MAC flush. At the receiver site, each PBB edge switch performs I-SID checking. If the I-SID matches one of the I-components, the MAC in the I-component is flushed.

The use of MIRP in 802.1ah networks is illustrated in the following figure.

Figure 46: MIRP in 802.1ah Networks



Device DHD1 is dual-homed to two 802.1ah backbone edge bridges (BEB1 and BEB2). Assume that initially the primary path is through BEB1. In this configuration BEB3 learns that the host behind DHD1 (with MAC address CM1) is reachable via the destination B-MAC M1. If the link between DHD1 and BEB1 fails and the host behind DHD1 remains inactive, the MAC cache tables on BEB3 still refer to the BEB1 MAC address even though the new path is now via BEB2 with B-MAC address M2. Any bridged traffic destined from the host behind DHD2 to the host behind DHD1 is wrongly encapsulated with B-MAC M1 and sent over the MAC tunnel to BEB1, where the traffic drops.

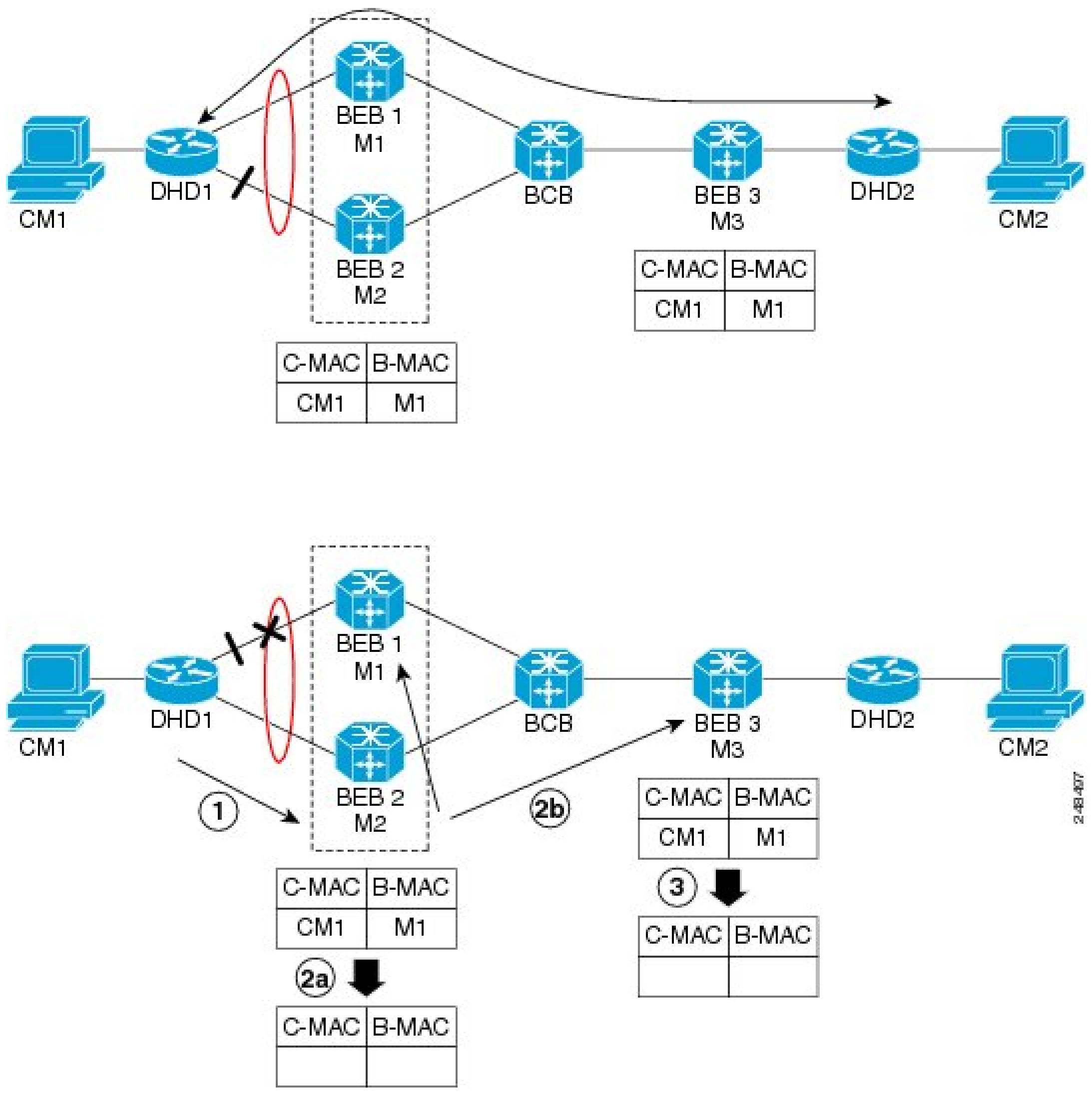
To circumvent the dropping of traffic when the link between DHD1 and BEB1 fails, BEB2 performs two tasks:

- Flushes it’s own MAC address table for the service or services.
- Requests the remote PE that receives the MIRP packet to clear it’s own MAC table. The MIRP message is transparent to the backbone core bridges (BCBs). The MIRP message is processed on a BEB because only BCBs learn and forward, based on B-MAC addresses and they are transparent to C-MAC addresses.

Note MIRP triggers C-MAC address flushing for both native 802.1ah and PBB over VPLS.

The following figure shows the operation of the MIRP.

Figure 47: MIRP Operation



Provider Backbone Bridging Ethernet VPN

The Provider Backbone Bridging Ethernet VPN (PBB-EVPN) is a next generation L2VPN solution that addresses resiliency and forwarding policy requirements. This feature also introduces advanced multihoming options, support for multipath and user-defined BGP policy capabilities to Ethernet L2VPNs. PBB-EVPN uses BGP for MAC address distribution and learning over the packet-switched network (PSN). PBB-EVPN is a combination of the capabilities of PBB and Ethernet VPN that addresses these Carrier Ethernet and data center interconnect requirements:

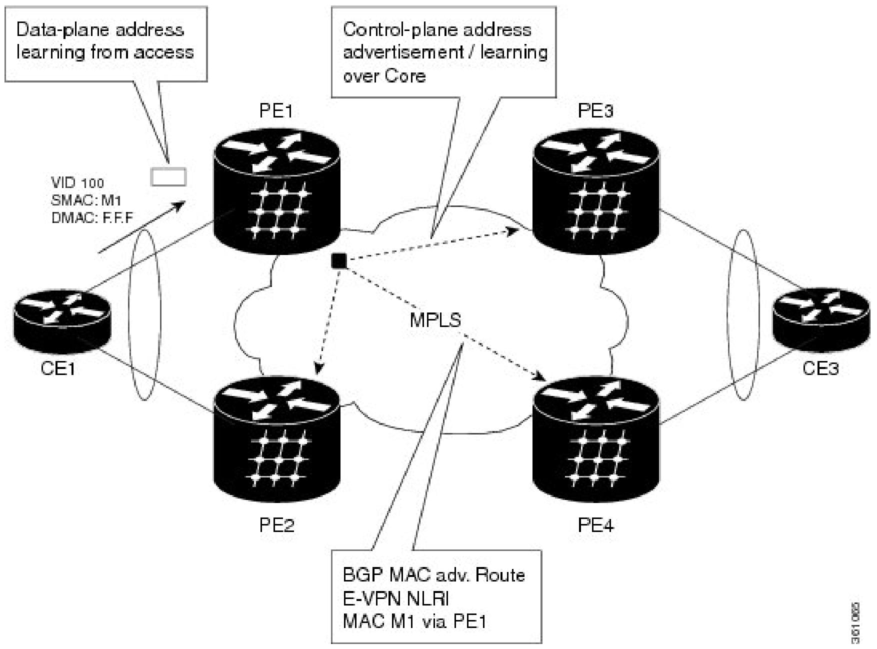
- All-active Redundancy and Load Balancing
- Simplified Provisioning and Operation
- Optimal Forwarding
- Fast Convergence
- MAC Address Scalability

Ethernet VPN

Ethernet Virtual Private Network (EVPN) is a solution for secure and private connectivity of multiple sites within an organization. The EVPN service extends the benefits of Ethernet technology to the Wide Area Network (WAN). This service is delivered over MPLS networks.

EVPN allows you to manage routing over a virtual private network, providing complete control and security. EVPN introduces a solution for multipoint L2VPN services, with advanced multi-homing capabilities, using BGP for distributing customer or client MAC address reachability information over the MPLS/IP network. EVPN advertises each customer MAC address as BGP routes, therefore allowing BGP policy control over MAC addresses.

Figure 48: MAC Distribution in BGP (EVPN)

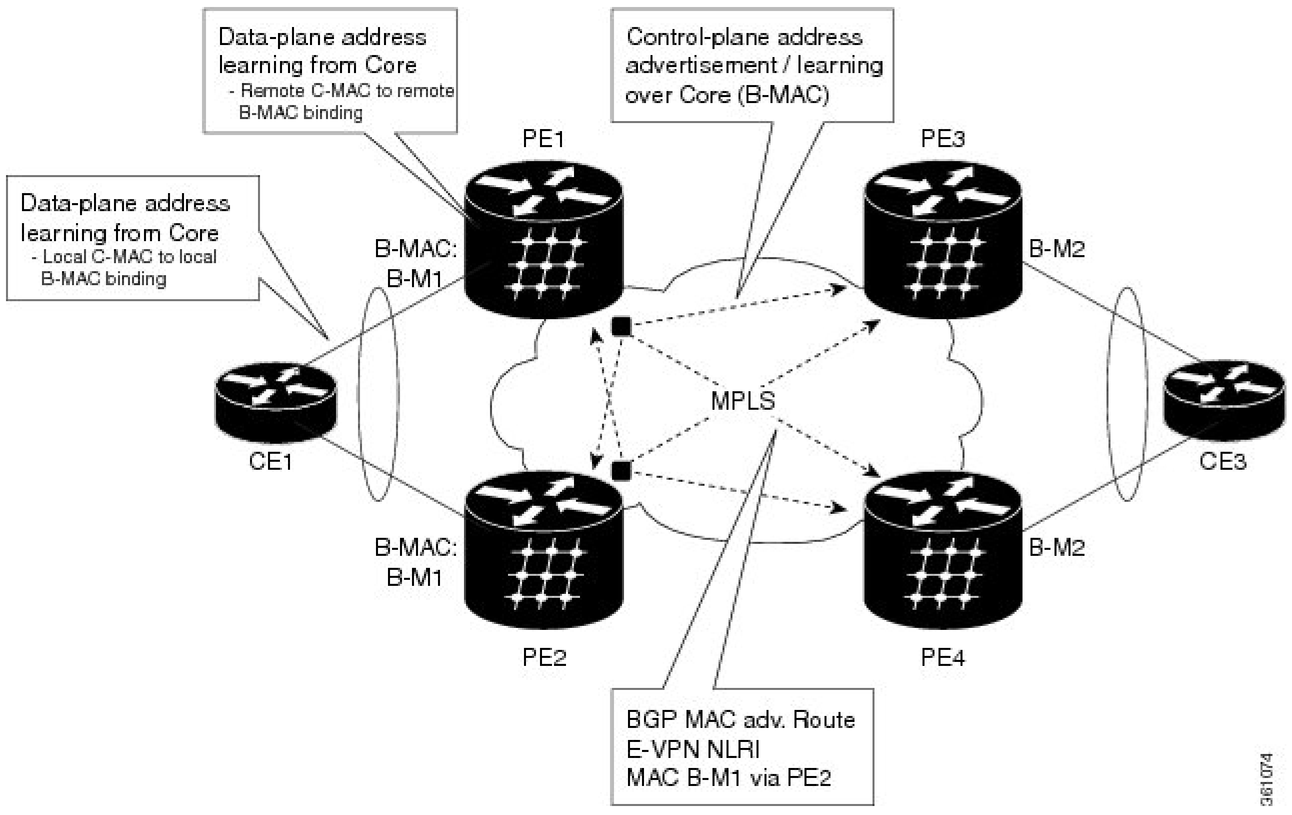


In the above figure, the provider edge (PE) routers run multi-protocol BGP to advertise and learn MAC addresses over MPLS. The customer MAC addresses are learnt in the data plane over attachment circuits (links connecting customer devices to the PEs). Then, the MAC addresses are distributed over MPLS using BGP with an MPLS label identifying the EVPN instance.

PBB-EVPN Overview

The PBB-EVPN solution combines Ethernet Provider Backbone Bridging (PBB - IEEE 802.1ah) with Ethernet VPN where, PEs perform as PBB Backbone Edge Bridge (BEB). The PEs receive 802.1Q Ethernet frames from their attachment circuits. These frames are encapsulated in the PBB header and forwarded over the IP/MPLS core. On the egress side (EVPN PE), the PBB header is removed after MPLS disposition, and the original 802.1Q Ethernet frame is delivered to the customer equipment.

Figure 49: PBB-EVPN Network


Data-plane address learning from Core
- Remote C-MAC to remote B-MAC binding
Control-plane address advertisement / learning over Core (B-MAC)
Data-plane address learning from Core
- Local C-MAC to local B-MAC binding
PE1
PE3
B-MAC: B-M1
B-M2
MPLS
CE1
CE3
B-MAC: B-M1
B-M2
PE2
PE4
BGP MAC adv. Route
E-VPN NLRI
MAC B-M1 via PE2
361074

The PE routers perform these functions:

- Learns customer or client MAC addresses (C-MACs) over the attachment circuits in the data-plane, per normal bridge operation.
- Learns remote C-MAC to backbone MAC (B-MAC) bindings in the data-plane from traffic ingress from the core.
- Advertises local B-MAC address reachability information in BGP to all other PE nodes in the same set of service instances. Note that every PE has a set of local B-MAC addresses that uniquely identify the device.
- Builds a forwarding table from the received remote BGP advertisements, associating remote B-MAC addresses with remote PE IP addresses.

PBB-EVPN scales well for large network with millions of customer MAC addresses by constraining customer MAC address in access. Only B-MAC addresses are advertised in core, making the number of BGP routes exchanged manageable.

For PBB EVPN, the B-MAC flush is per B-MAC per Ethernet VPN Instance (EVI).

EVPN Instance

E-VPN Instance (EVI) identifies a VPN in the MPLS/IP network. There can only be one EVI per core bridge.

Ethernet Segment

Ethernet Segment is a site connected to one or more PEs. The Ethernet Segment could be a single device (i.e. Customer Edge (CE)) or an entire network, such as:

- Single-Homed Device (SHD)
- Multi-Homed Device (MHD) using Ethernet Multi-chassis Link Aggregation Group
- Single-Homed Network (SHN)
- Multi-Homed Network (MHN)

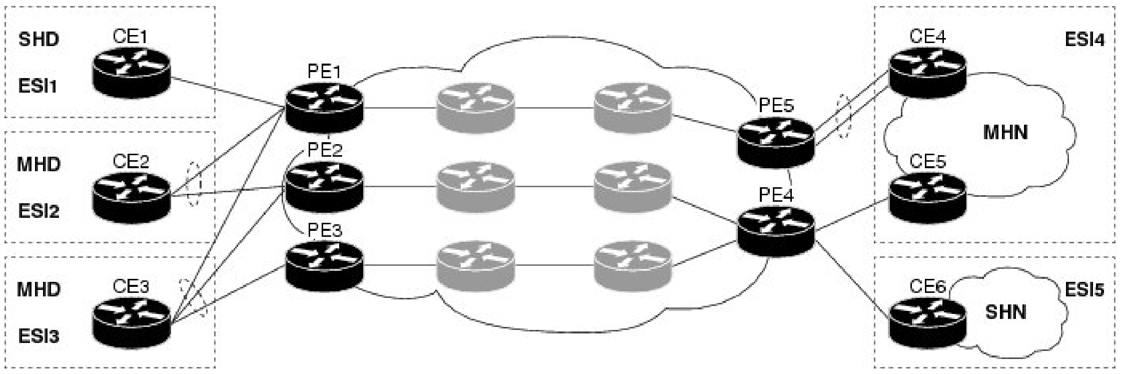
The Ethernet segment is uniquely identified by a 10-byte global Ethernet Segment Identifier (ESI).

The ESI format is RFC 7432 complaint. The ESI value depends on the ESI type. Currently, only ESI type 0 and 1 are supported. The following table shows the ESI format based on the ESI type.

ESI Type	Explanation	ESI Format
Type 0	Arbitrary ESI value based on configuration	1 octet ESI Type 0x00 9 octet ESI value
Type 1	Auto-generated ESI value based on LACP	1 octet ESI Type 0x01 6 octet CE LACP MAC address 2 octet CE LACP Port Key 1 octet value of 0x00

The following figure illustrates an example of Ethernet segment and ESI.

Figure 50: Ethernet Segment



PBB-EVPN BGP Routes

PBB-EVPN defines a single new BGP network layer reachability information (NLRI) used to advertise different types of routes along with new attributes.

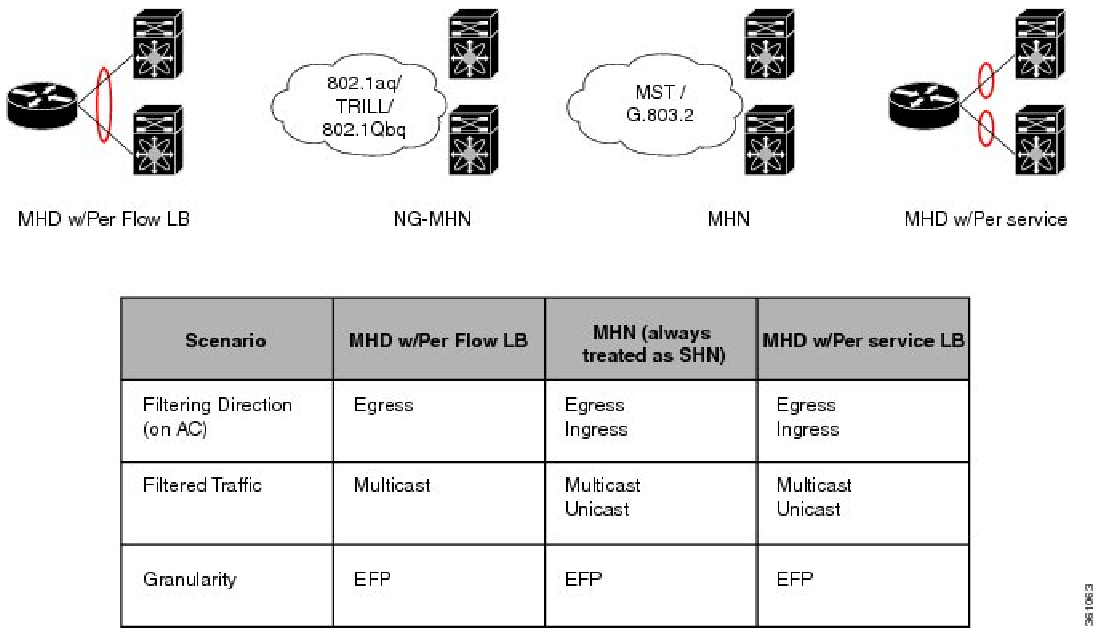
Designated Forwarder Election

The Designated Forwarder (DF) election mechanism is used to determine a designated forwarder in dual-homed or multi-homed devices or networks. The election is performed on a per service basis. The DF filtering function for MHN differs from that for MHD in:

- Directionality—DF filtering for MHN is applied for traffic both ingress and egress on the access-facing Ethernet interfaces; whereas, DF filtering for MHD is applied only to traffic that egress the access-facing interfaces.
- Traffic Type—DF filtering for MHN impacts both unicast as well as flooded multi-destination traffic; whereas, DF filtering for MHD only applies to flooded multi-destination traffic.

The following figure shows the various DF filtering rules for MHN and MHD.

Figure 51: DF Filtering Comparison for MHN/MHD



Scenario	MHD w/Per Flow LB	MHN (always treated as SHN)	MHD w/Per service LB
Filtering Direction (on AC)	Egress	Egress Ingress	Egress Ingress
Filtered Traffic	Multicast	Multicast Unicast	Multicast Unicast
Granularity	EFP	EFP	EFP

Access Auto-Sensing

PEs connected to a multi-homed or dual-homed device may support active-active per flow also known as flow-based load balancing. PE services CEs via physical or bundle ports. An Ethernet segment identifier is assigned per port. This value is calculated from the connected CE using information such as, CE system priority, CE system ID and CE port key. The PE must auto-detect the access topology to determine the type of load balancing. The load balancing could be active-active per flow load-balancing, per service load-balancing or simply no load balancing.

MMRP for PBB VPLS Flood Optimization

In a PBB network, traffic (unknown unicast, multicast, or broadcast) is flooded to all the PE devices in the network even if the devices do not host the service instance to which the traffic is destined.

The Multiple MAC Registration Protocol (MMRP) for PBB VPLS Flood Optimization feature optimizes the impact of the flooded traffic on PE devices by sending the traffic only to the PE devices interested in a particular service instance.

In a PBB over VPLS network, traffic between the PE devices flows over MPLS pseudo-wires that connect all the PE devices in a full mesh network topology.

Provider Back Bone Network Topology figure illustrates a typical 802.1ah PBB network.

For every I-SID (Service Instance VLAN ID) there is a corresponding multicast MAC address called the group B-MAC address, which is derived based on the I-SID. The group B-MAC address is used as the destination address in the outer MAC header when propagating flooded traffic across the provider backbone.

The MMRP is used by the PE devices to inform each other about the set of group B-MAC addresses corresponding to the I-SIDs of the service instances they host. This enables each device to determine which set of pseudo-wires flooded traffic should be forwarded on, that is, those pseudo-wires on which an MMRP registration has been received for the group B-MAC address corresponding to the I-SID.

Note The PBB-VPLS flood optimization feature is enabled only on PBB-VPLS network and not on PBB over Ethernet network.

Configuring PBB-VPLS Flood Optimization

To configure the PBB-VPLS flood optimization feature, do the following:

Enabling PBB-VPLS Flood Optimization on PBB Core Bridge

Perform this task to enable PBB-VPLS flood optimization on PBB core bridge.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **pbb core**
6. **mmrp-flood-optimization**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain pbb-core
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **pbb core**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb core
```

Configures the bridge domain as PBB core and enters the PBB core configuration submode.

This command also creates an internal port known as Customer bridge port (CBP).

All the interfaces (bridge ports) under this bridge domain are treated as the provider network ports (PNP).

Step 6 **mmrp-flood-optimization**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pbb-core)# mmrp-flood-optimization
```

Enables the flood optimization for PBB over VPLS feature on the core bridge.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Generic MRP Protocol Parameters

Perform this task to configure the generic MRP protocol parameters for PBB-VPLS flood optimization.

SUMMARY STEPS

1. **configure**
2. **mmrp-flood-optimization**
3. (Optional) **periodic transmit interval** *seconds*
4. (Optional) **join-time** *milliseconds*
5. (Optional) **leaveall-time** *seconds*
6. (Optional) **leave-time** *seconds*
7. (Optional) **flood-time** *seconds*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **mmrp-flood-optimization**

Example:

```
RP/0/RSP0/CPU0:router(config)# mmrp-flood-optimization
```

Enables the flood optimization for PBB over VPLS feature on the core bridge.

Step 3 (Optional) **periodic transmit interval** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mmrp-flood-opt)# periodic transmit interval 3
```

Enables periodic Multiple MAC Registration Protocol Data Units (MMRPDUs).

Step 4 (Optional) **join-time** *milliseconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mmrp-flood-opt)# join time interval 300
```

Sets the join time for all active ports.

Step 5 (Optional) **leaveall-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mmrp-flood-opt)# leaveall-time 10
```

Sets the leave all time for all active ports.

Step 6 (Optional) **leave-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mmrp-flood-opt)# leave-time 40
```

Sets the leave time for all active ports.

Step 7 (Optional) **flood-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mmrp-flood-opt)# flood-time 1000
```

Enables flooding of traffic to the entire core bridge when the PBB-VPLS Flood Optimization feature is enabled on the core bridge.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

How to Implement 802.1ah Provider Backbone Bridge

This section contains these procedures:

Restrictions for Implementing 802.1ah Provider Backbone Bridge

The following features are not supported:

- Cross-connect based point to point services over MAC-in-MAC
- One Edge bridge to multiple Core bridge mapping
- I type backbone edge bridge (I-BEB) and B type backbone edge bridge (B-BEB)

- IEEE 802.1ah over VPLS
- Multiple source B-MAC addresses per chassis
- Direct encapsulation of 802.1ah formatted packets natively over an MPLS LSP encapsulation
- To use the sub-interface configurations **encapsulation dot1ad** (or **encapsulation dot1q**) and **encapsulation dot1ad second-dot1q any** (or **encapsulation dot1q second-dot1q any**) together on the same physical interface, use the **exact** keyword as shown below. Else, it results in traffic loss.

```
Router(config)# interface hundredGigE 0/0/0/0.0
Router(config-subif)# encapsulation dot1ad 200 exact

Router(config)# interface hundredGigE 0/0/0/0.1
Router(config-subif)# encapsulation dot1ad 200 dot1q any
Router(config-subif)# commit
```

The following additional restriction applies when implementing Provider Backbone Bridge Ethernet VPN (PBB-EVPN):

- The Provider Edge and Route Reflector routers must run software supporting the same IETF draft version of L2VPN Ethernet VPN (EVPN). Due to the differences in BGP Network Layer Reachability Information (NLRI) encoding, later draft versions are not backward compatible with earlier ones. The following table shows the supported draft for various Cisco IOS XR software releases.

Cisco IOS XR software release	Supported L2VPN EVPN draft version	
	draft-ietf-l2vpn-evpn-04	draft-ietf-l2vpn-evpn-06
5.1.1 and older releases	✓	—
5.2.0	✓	—
5.1.2 and later releases except 5.2.0	—	✓

Configuring Ethernet Flow Points on CNP and PNP Ports

Perform this task to configure an Ethernet flow point (EFP) on the customer network port (CNP) or the provider network port (PNP).

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id.subinterface* **l2transport**
3. **encapsulation dot1q** *vlan-id* or **encapsulation dot1ad** *vlan-id* or **encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id*
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface** *type interface-path-id.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface
GigabitEthernet0/0/0/10.100 l2transport
```

Configures an interface for L2 switching.

Step 3 **encapsulation dot1q** *vlan-id* or **encapsulation dot1ad** *vlan-id* or **encapsulation dot1ad** *vlan-id* **dot1q** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-subif)# encapsulation dot1q 100
or
encapsulation dot1ad 100
or
encapsulation dot1ad 100 dot1q 101
```

Assigns the matching VLAN ID and Ethertype to the interface

Step 4 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring PBB Edge Bridge Domain and Service Instance ID

Perform this task to configure a PBB edge domain and the service ID.

Note To configure the PBB feature, login with admin user privileges and issue the **hw-module profile feature l2** command to select an ASR 9000 Ethernet line card ucode version that supports the PBB feature. The PBB feature will not be supported on the ASR 9000 Ethernet line card unless you make this configuration. For more information on configuring the feature profile, refer to the *Cisco ASR 9000 Series Aggregation Services Router System Management Configuration Guide*.

SUMMARY STEPS

1. **configure**
2. **l2vpn**

3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **interface** *type interface-path-id.subinterface*
6. **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain pbb-edge
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain, if it already exists.

Step 5 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/5/0/0.20
```

Assigns the matching VLAN ID and Ethertype to the interface. This EFP is considered as the CNP for the Edge bridge.

Step 6 **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb edge i-sid 1000 core-bridge pbb-core
```

Configures the bridge domain as PBB edge with the service identifier and the assigned core bridge domain, and enters the PBB edge configuration submode.

This command also creates the Virtual instance port (VIP) that associates the PBB Edge bridge domain to the specified Core bridge domain.

All the interfaces (bridge ports) under this bridge domain are treated as the customer network ports (CNP).

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the PBB Core Bridge Domain

Perform this task to configure the PBB core bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *group-name*
4. **bridge-domain** *domain-name*
5. **interface** *type interface-path-id.subinterface*
6. **pbb core**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group, if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain pbb-core
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/5/0/0.20
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 6 **pbb core**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb core
```

Configures the bridge domain as PBB core and enters the PBB core configuration submode.

This command also creates an internal port known as Customer bridge port (CBP).

All the interfaces (bridge ports) under this bridge domain are treated as the provider network ports (PNP).

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Backbone VLAN Tag under the PBB Core Bridge Domain

Perform this task to configure the backbone VLAN tag under the PBB core bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **interface** *type interface-path-id.subinterface*
6. **interface** *type interface-path-id. subinterface*
7. **pbb core**
8. **rewrite ingress tag push dot1ad** *vlan-id* **symmetric**
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain pbb-core
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/5/0/0.20
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 6 **interface** *type interface-path-id. subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# interface GigabitEthernet0/5/0/1.15
```

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. The interface now becomes an attachment circuit on this bridge domain.

Step 7 **pbb core**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb core
```

Configures the bridge domain as PBB core and enters the PBB core configuration submode.

This command also creates an internal port known as Customer bridge port (CBP).

All the interfaces (bridge ports) under this bridge domain are treated as the provider network ports (PNP).

Step 8 **rewrite ingress tag push dot1ad** *vlan-id* **symmetric**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pbb-core)# end
```

Configures the backbone VLAN tag in the Mac-in-MAC frame and also, sets the tag rewriting policy.

Note All PNPs in a Core bridge domain use the same backbone VLAN.

Step 9 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Backbone Source MAC Address

The backbone source MAC address (B-SA) is a unique address for a backbone network. Each Cisco ASR 9000 Series Router has one backbone source MAC address. If B-SA is not configured, then the largest MAC in the EEPROM is used as the PBB B-SA.

Note The backbone source MAC address configuration is optional. If you do not configure the backbone source MAC address, the Cisco ASR 9000 Series Routers allocate a default backbone source MAC address from the chassis backplane MAC pool.

Perform this task to configure the backbone source MAC address.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pbb**
4. **backbone-source-address** *mac-address*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **pbb**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# pbb
```

Enters PBB configuration mode.

Step 4 **backbone-source-address** *mac-address*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-pbb)# backbone-source-address 0045.1200.04
```

Configures the backbone source MAC address.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Unknown Unicast Backbone MAC under PBB Edge Bridge Domain

Perform this task to configure the unknown unicast backbone MAC under the PBB edge bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **interface** *type interface-path-id.subinterface*
6. **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*
7. **unknown-unicast-bmac** *mac-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RRP/0/RSP0/CPU0:router(config-l2vpn)# bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain pbb-edge
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/5/0/0.20
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 6 **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb edge i-sid 1000 core-bridge pbb-core
```

Configures the bridge domain as PBB edge with the service identifier and the assigned core bridge domain and enters the PBB edge configuration submode.

This command also creates the Virtual instance port (VIP) that associates the PBB Edge bridge domain to the specified Core bridge domain.

All the interfaces (bridge ports) under this bridge domain are treated as the customer network ports (CNP).

Step 7 **unknown-unicast-bmac** *mac-address*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pbb-edge)# unknown-unicast-bmac 1.1.1
```

Configures unknown unicast backbone MAC address.

Note On Trident line cards, once you configure the unknown unicast BMAC, the BMAC is used to forward customer traffic with multicast, broadcast and unknown unicast destination MAC address.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Static MAC addresses under PBB Edge Bridge Domain

Perform this task to configure the static MAC addresses under the PBB edge bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **interface** *type interface-path-id.subinterface*
6. **interface** *type interface-path-id.subinterface*
7. **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*
8. **static-mac-address** *cda-mac-address* **bmac** *bda-mac-address*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain pbb-edge
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface GigabitEthernet0/5/0/0.20
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 6 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#interface GigabitEthernet0/5/0/1.15
```

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. The interface now becomes an attachment circuit on this bridge domain.

Step 7 **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#pbb edge i-sid 1000 core-bridge pbb-core
```

Configures the bridge domain as PBB edge with the service identifier and the assigned core bridge domain and enters the PBB edge configuration submode.

This command also creates the Virtual instance port (VIP) that associates the PBB Edge bridge domain to the specified Core bridge domain.

All the interfaces (bridge ports) under this bridge domain are treated as the customer network ports (CNP).

Step 8 **static-mac-address** *cda-mac-address* **bmac** *bda-mac-address*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pbb-edge)#static-mac-address 0033.3333.3333 bmac 0044.4444.4444
```

Configures the static CMAC to BMAC mapping under the PBB Edge submode.

Step 9 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring PBB VPLS

Perform these tasks to configure PBB VPLS:

Configuring Access Pseudowire in I-Component

Perform this task to configure the static MAC addresses under the PBB edge bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **mac withdraw state-down**
6. **exit**
7. **interface** *type interface-path-id.subinterface*
8. **interface** *type interface-path-id.subinterface*
9. **neighbor** { *A.B.C.D* } **pw-id** *value*
10. **exit**
11. **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group pbb
```

Enters bridge group configuration mode. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain pbb-edge
```

Enters bridge domain configuration mode. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **mac withdraw state-down**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# mac withdraw state-down
```

(Optional) Enables MAC withdrawal.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-mac)# exit
```

Exits the current configuration mode.

Step 7 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/5/0/0.20
```

Assigns the matching VLAN ID and Ethertype to the interface.

Step 8 **interface** *type interface-path-id.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# interface GigabitEthernet0/5/0/1.15
```

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. The interface now becomes an attachment circuit on this bridge domain.

Step 9 **neighbor** { *A.B.C.D* } **pw-id** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#
```

Adds an access pseudowire port to a bridge domain.

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.

  Note *A.B.C.D* can be a recursive or non-recursive prefix.

- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 10 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)# exit
```

Exits the current configuration mode.

Step 11 **pbb edge i-sid** *service-id* **core-bridge** *core-bridge-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb edge i-sid 1000 core-bridge pbb-core
```

Configures the bridge domain as PBB edge with the service identifier and the assigned core bridge domain and enters the PBB edge configuration submode.

All the interfaces (bridge ports) under this bridge domain are treated as the customer network ports (CNP).

Step 12 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Core Pseudowire in B-Component

Perform this task to configure the static MAC addresses under the PBB edge bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *domain-name*
5. **vfi** { *vfi-name* }
6. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group pbb
```

Enters configuration mode for the named bridge group. This command creates a new bridge group or modifies the existing bridge group if it already exists. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain pbb-core
```

Enters configuration mode for the named bridge domain. This command creates a new bridge domain or modifies the existing bridge domain if it already exists.

Step 5 **vfi** { *vfi-name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi PBB-core-vfi
```

Configures the virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

- Use the *vfi-name* argument to configure the name of the specified virtual forwarding interface.

Step 6 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pw)#
```

Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.

  Note *A.B.C.D* can be a recursive or non-recursive prefix.

- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring PBB-EVPN

This section provides information on:

Configuring PBB Core Bridge Domains

Perform this task to create the PBB Core bridge domain and assign it's corresponding EVPN EVI ID.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *group_name*
4. **bridge-domain** *bridge_domain_name*
5. **pbb core**
6. **evpn evi** *evi_id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters L2VPN configuration mode.

Step 3 **bridge group** *group_name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group group1
```

Enters bridge group configuration mode. This command creates a new bridge group. A bridge group organizes bridge domains.

Step 4 **bridge-domain** *bridge_domain_name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain sample-pbb-core#
```

Enters bridge group domain configuration mode. This command creates a new bridge domain.

Step 5 **pbb core**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# pbb core
```

Configures the bridge domain as PBB core and enters the PBB core configuration submode.

This command also creates an internal port known as Customer bridge port (CBP). All the interfaces (bridge ports) under this bridge domain are treated as the provider network ports (PNP).

Step 6 **evpn evi** *evi_id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pbb-core)# evpn evi 100
```

Enters EVPN configuration mode and configures the Ethernet VPN ID. The EVI ID range is from 1 to 65534.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring PBB Edge Bridge Domains

As a pre-requisite, a PBB-EVPN provider edge (PE) must be configured with PBB Edge Bridge Domains which in one side are associated to ethernet flow points matching traffic from access interfaces and on the other side are linked to PBB Core Bridge Domains for traffic forwarding through the core.

For more information on configuring Edge Bridge Domains, see Configuring PBB Edge Bridge Domain and Service Instance ID.

Configuring EVPN Ethernet Segment

Explicit configuration of Ethernet Segment parameters such as ESI and service carving behaviors (manual or dynamic) is required only for Dual Homed scenarios with Active/Active per Service load-balancing.

Note By default, Dual Homed scenarios with Active/Active per Flow load-balancing auto-sense ESI values from CE's LACP information.

Note PBB-EVPN configuration allows to create only 24 ICCP-groups.

Perform this task to configure the EVPN Ethernet segment.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **interface type** *interface-path-id*
4. **ethernet-segment**
5. **backbone-source-mac** *mac_address*
6. **force single-homed**
7. **identifier type** *esi-type esi-identifier*
8. **bgp route-target** *ipv4/v6-address*
9. **load-balancing-mode per-service**
10. **service-carving manual primary** {*isid*} **secondary** {*isid*}
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# interface gigabitEthernet 0/1/0/4
```

Enters the physical port interface or the bundle interface configuration mode.

Step 4 **ethernet-segment**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
```

Enters the EVPN ethernet-segment configuration mode.

Step 5 **backbone-source-mac** *mac_address*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# backbone-source-mac 0045.1200.04
```

Configures the backbone source MAC address.

Step 6 **force single-homed**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# force single-homed
```

Specifies forced attributes for this Ethernet Segment.

Step 7 **identifier type** *esi-type esi-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 ce.01.ce.01.ce.01.ce.01.01
```

Configures the Ethernet segment identifier (ESI) of an interface.

Step 8 **bgp route-target** *ipv4/v6-address*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# bgp route-target ce01.ce01.ce01
```

Configures the BGP Import Route-Target for the Ethernet-Segment.

Note This command configuration is mandatory for ESI type 0.

Step 9 **load-balancing-mode per-service**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# load-balancing-mode per-service
```

Specifies the load balancing mode.

Step 10 **service-carving manual primary** {*isid*} **secondary** {*isid*}

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# service-carving manual primary 100 secondary 200
```

Specifies a list of service identifiers (isid) as active and standby services. The **isid** range is from 256 to 16777216.

Step 11 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring BGP Route Target

By default, these parameters are auto-derived from the PE's configuration:

- Route Distinguisher (RD) for global Ethernet Segment table

Default: Auto-generated RD based on loopback IP address

- EVI’s BGP Route Distinguisher (RD)

Default: Auto-generated RD based on loopback IP address

- EVI’s BGP Route Target. Default: Auto-generated RT based on EVI ID

Perform this task to overwrite the auto-generated BGP RD/RT values.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **bgp**
4. **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
5. **exit**
6. **evpn**
7. **evi** *evi_id*
8. **bgp**
9. **route-target** [**import** | **export**] { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
10. **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# bgp
```

Enters EVPN BGP configuration mode and configures static BGP settings for the Ethernet Segment ES:GLOBAL EVI, which is used for handling ES routes.

Step 4 **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-bgp)# rd 200:50
```

Configures the route distinguisher.

Step 5 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-bgp)# exit
```

Returns to the global configuration mode.

Step 6 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 7 **evi** *evi_id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# evi 100
```

Configures Ethernet VPN ID.

The EVI ID range is from 1 to 65534.

Step 8 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
```

Enters the BGP configuration mode for the specific EVI.

Step 9 **route-target** [**import** | **export**] { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target 10:20
```

Creates a route-target extended community.

- The import keyword imports routing information from the target VPN extended community.
- The export keyword exports routing information to the target VPN extended community.

Step 10 **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# rd 25:30
```

Configures the route distinguisher.

Step 11 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Global EVPN Timers

Perform this task to configure global EVPN timers.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **timers [flushagain | peering | programming | recovery]**
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router evpn
```

Enters EVPN configuration mode.

Step 3 **timers [flushagain | peering | programming | recovery]**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# timers flushagain 40
```

Configures global EVPN timers.

- Flush-again timer (for AApS only): When a MAC flush is sent, usually at the end of the programming timer expiration, a flush-again timer is started for the flush-again timer value. When it expires, another MAC flush message (MVRP or STP-TCN) is sent to the CE. This timer can be configured per segment-interface.

  Range: 0 to 120 seconds, 0 means disabled

  Default: 60 seconds

- Peering timer: Once all conditions are met to advertise to BGP, the PE waits for the peering timer value before advertising its RT, ESI and, Local MAC if it is Single-Home.

  Range: 0 to 300 seconds, 0 means disabled

  Default: 45 seconds

- Programming timer: Indicated time required by the HW to apply the carving results. At the end of the programming timer expiration, the next Ethernet Segment route object will be processed.

  Range: 0 to 100000 microseconds

  Default: 1500 microseconds

- Recovery timer (for AApS only): Once the interface is up, the PE waits for the recovery timer value in order to allow the CE running STP protocol to converge. This timer can be configured per segment-interface.

  Range: 20 to 3600 seconds

  Default: 20 seconds

Note Changing timers is only useful for scale configurations.

Step 4 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring EVPN Timers Per Ethernet Segment and CE flushing mechanism

Perform this task to configure per Ethernet segment timers.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **interface type** *interface-path-id*
4. **ethernet-segment**
5. **mac-flush mvrp**
6. **timers [flushagain | recovery]**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 **interface type** *interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# interface gigabitEthernet 0/1/0/4
```

Enters the physical port interface or the bundle interface configuration mode.

Step 4 **ethernet-segment**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
```

Enters the EVPN ethernet-segment configuration mode.

Step 5 **mac-flush mvrp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)#
```

Specifies MAC flush mode for this Ethernet Segment.

Step 6 **timers [flushagain | recovery]**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# timers flushagain 40
```

Configures per Ethernet segment timers.

- Flush-again timer (for AApS only): When a MAC flush is sent, usually at the end of the programming timer expiration, a flush-again timer is started for the flush-again timer value. When it expires, another MAC flush message (MVRP or STP-TCN) is sent to the CE. This timer can be configured per segment-interface.

  Range: 0 to 120 seconds, 0 means disabled

  Default: 60 seconds

- Recovery timer (for AApS only): Once the interface is up, the PE waits for the recovery timer value in order to allow the CE running STP protocol to converge. This timer can be configured per segment-interface.

  Range: 20 to 3600 seconds

  Default: 20 seconds

Note Changing timers is only useful for scale configurations.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Multichassis Link Aggregation

Multichassis Link Aggregation (MCLAG) is used in scenarios involving Multi Homed Devices. You must create an ICCP redundancy group in order to specify relevant MLACP parameters, such as, **mlacp system mac, mlacp system priority, mlacp node id** and backbone interfaces.

Note Even though the redundancy group is created under the **redundancy-iccp-group** sub-mode, the solution does not rely on an actual ICCP session between PEs connected to the same site. The mode singleton command has been introduced to alert ICCP module.

For more information on configuring MCLAG, refer to the Configuring Link Bundling on the Cisco ASR 9000 Series Router module in the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide*.

Configuring BGP Routing Process

A prerequisite of PBB-EVPN involves enabling the new EVPN address family under the BGP routing process and under BGP neighbor submode. For more information on BGP, refer to the *Implementing BGP* module in the *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*.

Perform this task to enable EVPN address family under BGP routing process and BGP neighbor submode.

SUMMARY STEPS

1. **configure**
2. **router bgp** *asn_id*
3. **address-family l2vpn evp**
4. **exit**
5. **neighbor** *peer_ip_add*
6. **address-family l2vpn evpn**
7. **address-family l2vpn evpn**
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp** *asn_id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family l2vpn evp**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
```

Enables EVPN address family under BGP routing process and enters EVPN address family configuration submode.

Step 4 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# exit
```

Exits the current configuration mode.

Step 5 **neighbor** *peer_ip_add*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.1.1.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 6 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# exit
```

Enables EVPN address family under BGP routing process and enters EVPN address family configuration submode.

Step 7 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
```

Enables EVPN address family under BGP routing process and enters EVPN address family configuration submode.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

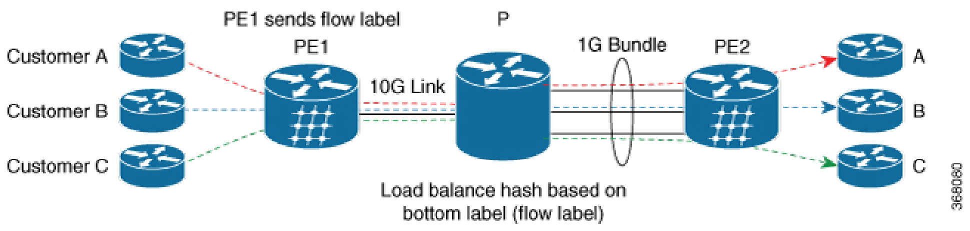
- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

PBB EVPN Flow Label

The PBB EVPN Flow Label feature enables provider (P) routers to use flow-based load balancing to forward traffic between the provider edge (PE) devices. A flow label is created based on indivisible packet flows entering an imposition PE, and it is inserted as the lower most label in the packet. P routers use flow label for load balancing to provide better traffic distribution across ECMP paths or link-bundled paths in the core. A flow is identified either by the source and destination IP address and layer 4 source and destination ports of the traffic, or the source and destination MAC address of the traffic.

Consider the following topology where PBB EVPN imposition router, PE1, adds a flow label in the EVPN traffic. The PBB EVPN disposition router, PE2, allows mixed types of traffic; some have flow label, others do not. The P router use flow label to load balance the traffic between the PEs . PE2 ignores flow label in traffic, and uses one EVPN label for all unicast traffic.

Figure 52: PBB EVPN Flow Label



Configure PBB EVPN Flow Label

Perform these tasks to configure PBB EVPN Flow Label feature.

```
RP/0/RSP0/CPU0:router#configure
RP/0/RSP0/CPU0:router(config)#l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)#bridge group PBB
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain EDGE
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#interface Bundle-Ether1.10
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)#pbb edge i-sid 1010 core-bridge CORE
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac-pbb-edge)#exit
!
RP/0/RSP0/CPU0:router(config-l2vpn-bg)#bridge-domain CORE
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)#pbb-core
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-pbb-core)#evpn evi 10
!
RP/0/RSP0/CPU0:router(config)#evpn
RP/0/RSP0/CPU0:router(config-evpn)#evi 10
RP/0/RSP0/CPU0:router(config-evpn-instance)#load-balancing
RP/0/RSP0/CPU0:router(config-evpn-instance-lb)#flow-label static
!

RP/0/RSP0/CPU0:router(config)#router bgp 20
RP/0/RSP0/CPU0:router(config-bgp)#address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-af)#commit
```

Running Configuration

This section shows PBB EVPN Flow Label running configuration.

```
l2vpn
 bridge group PBB
  bridge-domain EDGE
   interface Bundle-Ether1.10
   pbb edge i-sid 1010 core-bridge CORE

  bridge-domain CORE
   pbb-core
     evpn evi 10

evpn
 evi 10
  load-balancing
   flow-label static
```

```
router bgp 20
address-family l2vpn evpn
```

Verification

Verify PBB EVPN Flow Label configuration.

```
RP/0/RSP0/CPU0:router#show evpn evi vpn-id 10 detail
EVI        Bridge Domain                Type
---------- ---------------------------- -------
10         EVPN                         PBB
   Unicast Label  : 24001
   Multicast Label: 24002
   Flow Label: Y
   RD Config: none
   RD Auto  : (auto) 1.1.1.1:10
   RT Auto  : 1:10
   Route Targets in Use           Type
   ------------------------------ -------
   1:10                           Import
   1:10                           Export
```

Configuration Examples for Implementing 802.1ah Provider Backbone Bridge

This section provides these configuration examples:

Configuring Ethernet Flow Points: Example

This example shows how to configure Ethernet flow points:

```
config
interface GigabitEthernet0/0/0/10.100 l2transport
 encapsulation dot1q 100
or
 encapsulation dot1ad 100
or
 encapsulation dot1ad 100 dot1q 101
```

Configuring PBB Edge Bridge Domain and Service Instance ID: Example

This example shows how to configure the PBB edge bridge domain:

```
config
l2vpn
 bridge group PBB
  bridge-domain PBB-EDGE
  interface GigabitEthernet0/0/0/38.100
  !
  interface GigabitEthernet0/2/0/30.150
  !
  pbb edge i-sid 1000 core-bridge PBB-CORE
    !
 !
!
```

Configuring PBB Core Bridge Domain: Example

This example shows how to configure the PBB core bridge domain:

```
config
l2vpn
 bridge group PBB
  bridge-domain PBB-CORE
  interface G0/5/0/10.100
  !
  interface G0/2/0/20.200
  !
   pbb core
   !
 !
!
```

Configuring Backbone VLAN Tag: Example

This example shows how to configure the backbone VLAN tag:

```
config
l2vpn
 bridge group PBB
  bridge-domain PBB-CORE
  interface G0/5/0/10.100
  !
  interface G0/2/0/20.200
  !
   pbb core
    rewrite ingress tag push dot1ad 100 symmetric
   !
 !
!
```

Configuring Backbone Source MAC Address: Example

This example shows how to configure the backbone source MAC address:

```
config
l2vpn
 pbb
  backbone-source-mac 0045.1200.04
 !
!
```

Configuring Static Mapping and Unknown Unicast MAC Address under the PBB Edge Bridge Domain

This example shows how to configure static mapping and unknown unicast MAC address under the PBB edge bridge domain:

```
config
l2vpn
 bridge group PBB
  bridge-domain PBB-EDGE
  interface GigabitEthernet0/0/0/38.100
  !
  interface GigabitEthernet0/2/0/30.150
  !
  pbb edge i-sid 1000 core-bridge PBB-CORE
   static-mac-address 0033.3333.3333 bmac 0044.4444.4444
   unknown-unicast-bmac 0123.8888.8888
   !
 !
!
```

Configuring PBB-VPLS: Example

This example shows you how to configure PBB VPLS.

Configuring Access Pseudowire in I-component

```
l2vpn
  bridge group PBB
    bridge-domain PBB-EDGE
      mac withdraw state-down ------ can be used with MIRP, optional
      interface GigabitEthernet0/0/0/38.100
      interface GigabitEthernet0/2/0/30.150
      neighbor 10.10.10.1 pw-id 1010 ------- configures access PW
      !
      pbb edge i-sid 1200 core-bridge PBB-CORE
   !
 !
!
```

Configuring Core Pseudowire in B-component

```
l2vpn
  bridge group PBB
    bridge-domain PBB-CORE
      interface G0/5/0/10.100
      !
    vfi PBB-CORE-vfi
     neighbor 1.1.1.1 pw-id 1004 ------- configures core PW
     !
!
 !
```

Configuring MIRP Lite: Example

The MIRP feature is enabled by default. However, MIRP packets are sent when the attachment circuit is not functional and you have configured **mac withdraw state-down** as shown:

```
l2vpn
  bridge group PBB
    bridge-domain PBB-EDGE
```

mac withdraw state-down

However, if you have not configured mac withdraw state-down, then MIRP packets are sent when the attachment circuit is functional.

Configuring PBB-EVPN: Example

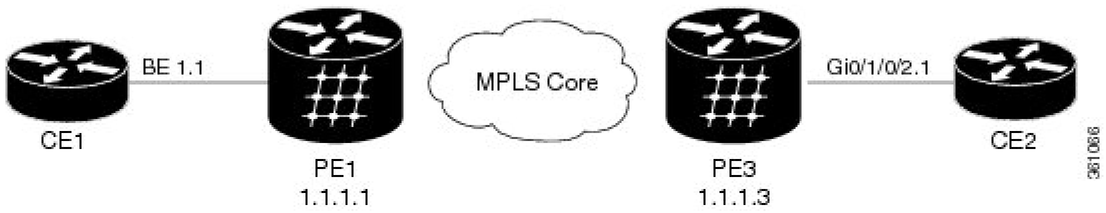
This section provides examples for:

PBB-EVPN on Single Homed Device/Single Homed Network

This example covers:

- PBB-EVPN service between two PEs in the same AS with single homed CEs
- Dual attached CE using a bundle interface connected to PE1.
- Single attached CE connected to PE2
- EVI carrying traffic from single I-SID
- PBB source MAC customized via configuration on both PEs for easier tracking
- BGP RD/RT auto-derived from BGP ASN and EVI ID

Figure 53: PBB-EVPN on Single Homed Device/Single Homed Network



Configuration on PE1:

```
interface Bundle-Ether1.1 l2transport
   encapsulation dot1q 1 200

l2vpn
  pbb
    backbone-source-mac 00aa.00bb.00cc
  bridge group gr1
    bridge-domain bd1
      interface Bundle-Ether1.1
      pbb edge i-sid 300 core-bridge core_bd1

  bridge group gr2
      bridge-domain core_bd1
        pbb core
          evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.1
 address-family l2vpn evpn
 !
 neighbor 1.1.1.3
  remote-as 100
  address-family l2vpn evpn
```

Configuration on PE3:

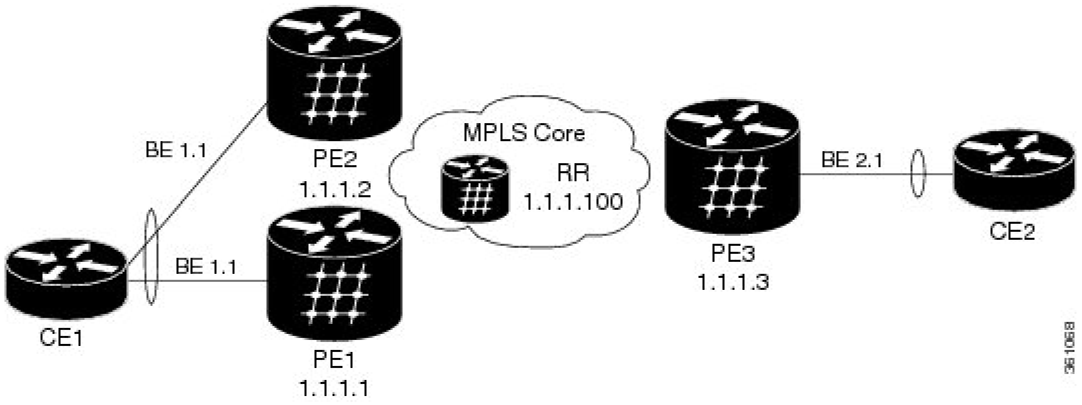
```
interface GigabitEthernet 0/1/0/2.1 l2transport
   encapsulation dot1q 200

l2vpn
  pbb
    backbone-source-mac 00bb.00cc.00dd
  bridge group gr1
    bridge-domain bd1
      interface GigabitEthernet0/1/0/2.1
      pbb edge i-sid 300 core-bridge core_b1

  bridge group gr2
    bridge-domain core_bd1
      pbb core
        evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.3
 address-family l2vpn evpn
 !
 neighbor 1.1.1.1
  remote-as 100
  address-family l2vpn evpn
```

PBB EVPN on Dual Homed Device/Multi Homed Device with Active/Active per Flow load-balancing

This examples covers:

- PBB-EVPN service among three PEs in the same AS with a dual homed CE (behind PE1 and PE2) and a single homed CE (behind PE3)
- PE1/PE2 configured to perform active/active per Flow loadbalancing allowing ingress traffic from the same I-SID to be handled by both PEs
- Example shows EVI carrying traffic from a single I-SID
- PBB I-SID values must match among PEs connected to a common dual homed site
- ICCP must be configured in PE1, PE2 using a new mode (mode singleton); where no ICCP neighbor is configured. Note that MLACP parameters such as system MAC/priority must be identical while MLACP node ID must be unique on PE1/PE2
- ESI must be identical on PEs connected to a common dual homed site. Example shows default behavior where ESI value is auto-derived from CE's LACP information
- PBB source MAC must be the same on PEs connected to a dual homed site operating on active/active per flow load-balancing. Example shows default behavior where PBB source MAC value is auto-derived from CE's LACP information
- CE must be configured with one bundle interface that includes all member interfaces connecting to both PEs
- BGP RD/RT auto-derived from BGP ASN and EVI ID

Figure 54: PBB EVPN on Dual Homed Device/Multi Homed Device with Active/Active per Flow load-balancing



Configuration on PE1:

```
redundancy
 iccp
  group 1
   mlacp node 1
   mlacp system mac 0aaa.0bbb.0ccc
   mlacp system priority 1
   backbone interface GigabitEthernet0/1/0/2
   mode singleton

interface bundle-Ether1
mlacp iccp-group 1

interface bundle-Ether1.1 l2transport
  encapsulation dot1q  10
l2vpn
  bridge group gr1
    bridge-domain bd1
      interface bundle-ether1.1
      pbb edge i-sid 600 core-bridge core_bd1

    bridge group gr2
      bridge-domain core_bd1
        pbb core
          evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.1
 address-family l2vpn evpn
 !
 neighbor 1.1.1.100
  remote-as 100
  address-family l2vpn evpn
```

Configuration on PE2:

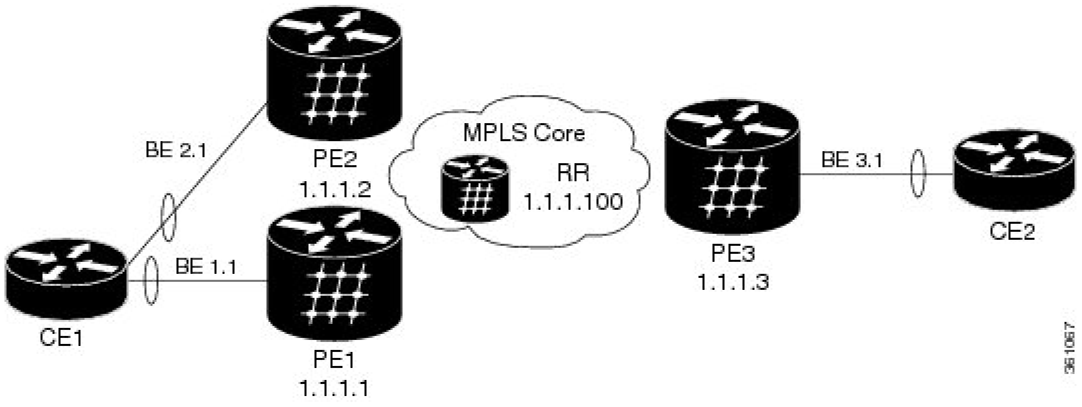
```
redundancy
 iccp
  group 1
   mlacp node 2
   mlacp system mac 0aaa.0bbb.0ccc
   mlacp system priority 1
   backbone interface GigabitEthernet0/1/0/2
   mode singleton

interface bundle-Ether1
mlacp iccp-group 1

interface bundle-Ether1.1 l2transport
  encapsulation dot1q  10
l2vpn
  bridge group gr1
    bridge-domain bd1
      interface bundle-Ether1.1
      pbb edge i-sid 600 core-bridge core_b1

    bridge group gr2
      bridge-domain core_bd1
        pbb core
            evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.2
 address-family l2vpn evpn
 !
 neighbor 1.1.1.100
  remote-as 100
  address-family l2vpn evpn
```

PBB EVPN on Dual Homed Device/Multi Homed Device with Active/Active per Service load-balancing and Dynamic Service Carving

This example covers:

- PBB-EVPN service among three PEs in the same AS with a dual homed CE (behind PE1 and PE2) and a single homed CE (behind PE3)
- PE1/PE2 configured to perform active/active per service (i.e. per-ISID) loadbalancing with dynamic service carving/DF election allowing traffic from some I-SIDs to be handled by PE1 while the rest to be handled by PE2
- EVI carrying traffic from two I-SIDs
- PBB I-SID values must match among PEs connected to a common dual homed site
- ICCP must be configured in PE1, PE2 using a new mode (mode singleton); where no ICCP neighbor is configured. ICCP configuration required to handle core isolation failures. Example uses same MLACP system mac/priority and unique MLACP node values on PE1/PE2
- ESI must be identical between PEs for a dual homed site. User configuration must be entered to guarantee that
- PBB source MAC must be different on each PE connected to a dual homed site. By default, PE uses system-wide PBB source MAC
- CE must be configured with two bundle interfaces. One for each set of member interfaces leading to a different PE
- BGP RD/RT auto-derived from BGP ASN and EVI ID

Figure 55: PBB EVPN on Dual Homed Device/Multi Homed Device with Active/Active per Service load-balancing and Dynamic Service Carving



```
Configuration on PE1:

redundancy
 iccp
  group 66
   mlacp node 1
   mlacp system mac 0aaa.0bbb.0ccc
   mlacp system priority 1
   backbone interface GigabitEthernet0/1/0/2
   mode singleton

interface Bundle-Ether1
 mlacp iccp-group 66
interface bundle-Ether1.1 l2transport
   encapsulation dot1q 10

interface bundle-Ether1.20 l2transport
   encapsulation dot1q 20

evpn
  interface bundle-Ether1
    ethernet-segment
       identifier type 0 01.11.00.00.00.00.00.00.01
      load-balancing-mode per-service

l2vpn
  bridge group gr1
    bridge-domain bd1
      interface bundle-ether1.1
      pbb edge i-sid 300 core-bridge core_bd1

    bridge-domain bd20
      interface bundle-ether1.20
      pbb edge i-sid 320 core-bridge core_bd1

   bridge group gr2
     bridge-domain core_bd1
            pbb core
                evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.1
 address-family l2vpn evpn
 !
 neighbor 1.1.1.100
  remote-as 100
  address-family l2vpn evpn
```

Configuration on PE2:

```
redundancy
 iccp
  group 66
   mlacp node 2
   mlacp system mac 0aaa.0bbb.0ccc
   mlacp system priority 1
   backbone interface GigabitEthernet0/1/0/2
   mode singleton

interface Bundle-Ether2
 mlacp iccp-group 66

interface bundle-Ether2.1 l2transport
   encapsulation dot1q 10


interface bundle-Ether2.20 l2transport
   encapsulation dot1q 20


evpn
  interface bundle-Ether2
    ethernet-segment
      identifier type 0 01.11.00.00.00.00.00.00.01
      load-balancing-mode per-service
l2vpn
  bridge group gr1
    bridge-domain bd1
      interface bundle-Ether2.1
      pbb edge i-sid 300 core-bridge core_bd1

    bridge-domain bd20
      interface bundle-Ether2.20
      pbb edge i-sid 320 core-bridge core_bd1

  bridge group gr2
         bridge-domain core_bd1
              pbb core
                   evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.2
 address-family l2vpn evpn
 !
 neighbor 1.1.1.100
  remote-as 100
  address-family l2vpn evpn
```

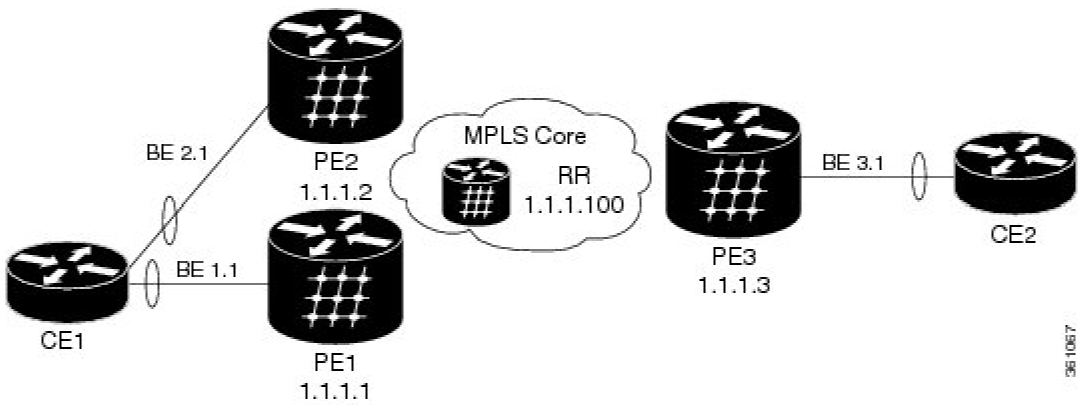
PBB EVPN on Dual Homed Device/Multi Homed Device with Active/Active per Service load-balancing and Manual Service Carving

This example covers:

- PBB-EVPN service among three PEs in the same AS with a dual homed CE (behind PE1 and PE2) and a single homed CE (behind PE3)

- PE1/PE2 configured to perform active/active per service (i.e. per-ISID) load balancing with manual service carving/DF election
- PE1 configured to forward traffic from I-SID range 256-276 and backup for I-SID 277-286. PE2 configured to behave in the opposite manner of PE1
- EVI carrying traffic from two I-SIDs
- PBB I-SID values must match among PEs connected to a common dual homed site
- ICCP must be configured in PE1, PE2 using a new mode (mode singleton); where no ICCP neighbor is configured. ICCP configuration required to handle core isolation failures. Example uses same MLACP system mac/priority and unique MLACP node values on PE1/PE2
- ESI must be identical between PEs for a dual homed site. User configuration must be entered to guarantee that
- PBB source MAC must be different on each PE connected to a dual homed site. Example below customizes PBB source MAC value via configuration for easier tracking
- CE must be configured with two bundle interfaces. One for each set of member interfaces leading to a different PE
- BGP RD/RT auto-derived from BGP ASN and EVI ID

Figure 56: PBB EVPN on Dual Homed Device/Multi Homed Device with Active/Active per Service load-balancing and Manual Service Carving



Configuration on PE1:

```
redundancy
 iccp
  group 66
   mlacp node 1
   mlacp system mac 0aaa.0bbb.0ccc
   mlacp system priority 1
   backbone interface GigabitEthernet0/1/0/2
   mode singleton

interface Bundle-Ether1
 mlacp iccp-group 66

interface bundle-Ether1.1 l2transport
  encapsulation dot1q  10
```

```
interface bundle-Ether1.20 l2transport
  encapsulation dot1q 20

evpn
  interface bundle-Ether1
    ethernet-segment
      identifier type 0 01.11.00.00.00.00.00.00.01
      load-balancing-mode per-service
      service-carving manual primary isid 256-276 secondary isid 277-286
l2vpn
  pbb
    backbone-source-mac 00aa.00bb.00cc
  bridge group gr1
    bridge-domain bd_256
      interface bundle-ether1.1
      pbb edge i-sid 260 core-bridge core_bd1

    bridge-domain bd_286
      interface bundle-ether1.20
      pbb edge i-sid 280 core-bridge core_bd1
  bridge group gr2
    bridge-domain core_bd1
      pbb core
        evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.1
 address-family l2vpn evpn
 !
 neighbor 1.1.1.100
  remote-as 100
  address-family l2vpn evpn
```

Configuration on PE2:

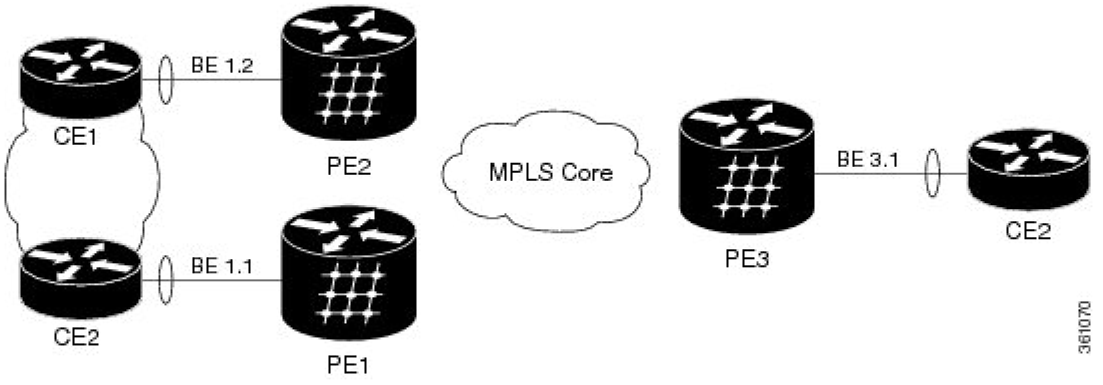
```
redundancy
 iccp
  group 66
   mlacp node 2
   mlacp system mac 0aaa.0bbb.0ccc
   mlacp system priority 1
   backbone interface GigabitEthernet0/1/0/2
   mode singleton

interface Bundle-Ether2
 mlacp iccp-group 66

interface bundle-Ether2.1 l2transport
  encapsulation dot1q  10

interface bundle-Ether2.20 l2transport
  encapsulation dot1q 20


evpn
  interface bundle-Ether2
    ethernet-segment
      identifier type 0 01.11.00.00.00.00.00.00.01
        load-balancing-mode per-service
        service-carving manual primary 277-286 secondary 256-276

l2vpn
  pbb
```

```
    backbone-source-mac 00cc.00dd.00ee
  bridge group gr1
    bridge-domain bd1
      interface bundle-Ether2.1
      pbb edge i-sid 260 core-bridge core_b1

    bridge-domain bd30

      Interface bundle-Ether2.20
      pbb edge i-sid 280 core-bridge core_b1
  bridge group gr2
    bridge-domain core_bd1
      pbb core
        evpn evi 1000
!
router bgp 100
 bgp router-id 1.1.1.2
 address-family l2vpn evpn
 !
 neighbor 1.1.1.100
  remote-as 100
  address-family l2vpn evpn
```

PBB-EVPN Multi Homed Network

This example shows how to configure PBB-EVPN on a Multi Homed Network with Active-Active per service load balancing:

Figure 57: PBB-EVPN Multi Homed Network



Configuration on PE1:

```
interface bundle-Ether1.1 l2transport
  encapsulation dot1q  1

evpn
  interface bundle-Ether1
    ethernet-segment
        load-balancing-mode per-service
l2vpn
  pbb
    backbone-source-mac 00aa.00bb.00cc
  bridge group gr1
    bridge-domain bd1
      interface bundle-ether1.1
      pbb edge i-sid 400 core-bridge core_bd1
    bridge group gr2
      bridge-domain core_bd1
        pbb core
          evpn evi 1000
```

Configuration on PE2:

```
interface bundle-Ether1.1 l2transport
  encapsulation dot1q  1
evpn
  interface bundle-Ether1
    ethernet-segment
      load-balancing-mode per-service

l2vpn
  pbb
    backbone-source-mac 00cc.00dd.00ee
  bridge group gr1
    bridge-domain bd1
      interface bundle-Ether1.1
      pbb edge i-sid 400 core-bridge core_bd1

  bridge group gr2
    bridge-domain core_bd1
      pbb core
        evpn evi 1000
```

CHAPTER 8

Implementing Multiple Spanning Tree Protocol

This module provides conceptual and configuration information for Multiple Spanning Tree Protocol on Cisco ASR 9000 Series Routers. Multiple Spanning Tree Protocol (MSTP) is a spanning-tree protocol used to prevent loops in bridge configurations. Unlike other types of STPs, MSTP can block ports selectively by VLAN.

Release	Modification
Release 3.7.3	This feature was introduced on Cisco ASR 9000 Series Routers
Release 3.9.1	Support for MSTP over Bundles feature was added.
Release 4.0.1	Support for PVST+ and PVSTAG features was added.
Release 4.1.0	Support for MSTAG Edge Mode feature was added.
Release 4.3.0	Support for PVSTAG was added on Bundle interfaces.
Release 5.1.0	Support for PVRST was added.

Prerequisites for Implementing Multiple Spanning Tree Protocol

This prerequisite applies to implementing MSTP:

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing Multiple Spanning Tree Protocol

To implement Ethernet services access lists, you must understand these concepts:

Spanning Tree Protocol Overview

Ethernet is no longer just a link-layer technology used to interconnect network vehicles and hosts. Its low cost and wide spectrum of bandwidth capabilities coupled with a simple *plug and play* provisioning philosophy have transformed Ethernet into a legitimate technique for building networks, particularly in the access and aggregation regions of service provider networks.

Ethernet networks lacking a TTL field in the Layer 2 (L2) header and, encouraging or requiring multicast traffic network-wide, are susceptible to broadcast storms if loops are introduced. However, loops are a desirable property as they provide redundant paths. Spanning tree protocols (STP) are used to provide a loop free topology within Ethernet networks, allowing redundancy within the network to deal with link failures.

There are many variants of STP; however, they work on the same basic principle. Within a network that may contain loops, a sufficient number of interfaces are disabled by STP so as to ensure that there is a loop-free spanning tree, that is, there is exactly one path between any two devices in the network. If there is a fault in the network that affects one of the active links, the protocol recalculates the spanning tree so as to ensure that all devices continue to be reachable. STP is transparent to end stations which cannot detect whether they are connected to a single LAN segment or to a switched LAN containing multiple segments and using STP to ensure there are no loops.

STP Protocol Operation

All variants of STP operate in a similar fashion: STP frames (known as bridge protocol data units (BPDUs)) are exchanged at regular intervals over Layer 2 LAN segments, between network devices participating in STP. Such network devices do not forward these frames, but use the information to construct a loop free spanning tree.

The spanning tree is constructed by first selecting a device which is the *root* of the spanning tree (known as the root bridge), and then by determining a loop free path from the *root bridge* to every other device in the network. Redundant paths are disabled by setting the appropriate ports into a blocked state, where STP frames can still be exchanged but data traffic is never forwarded. If a network segment fails and a redundant path exists, the STP protocol recalculates the spanning tree topology and activates the redundant path, by unblocking the appropriate ports.

The selection of the root bridge within a STP network is determined by the lowest Bridge ID which is a combination of configured bridge priority and embedded mac address of each device. The device with the lowest priority, or with equal lowest priority but the lowest MAC address is selected as the root bridge.

The selection of the active path among a set of redundant paths is determined primarily by the port path cost. The port path cost represents the cost of transiting between that port and the root bridge - the further the port is from the root bridge, the higher the cost. The cost is incremented for each link in the path, by an amount that is (by default) dependent on the media speed. Where two paths from a given LAN segment have an equal cost, the selection is further determined by the lowest bridge ID of the attached devices, and in the case of two attachments to the same device, by the configured port priority and port ID of the neighboring attached ports.

Once the active paths have been selected, any ports that do not form part of the active topology are moved to the blocking state.

Topology Changes

Network devices in a switched LAN perform MAC learning; that is, they use received data traffic to associate unicast MAC addresses with the interface out of which frames destined for that MAC address should be sent. If STP is used, then a recalculation of the spanning tree (for example, following a failure in the network) can invalidate this learned information. The protocol therefore includes a mechanism to notify topology changes around the network, so that the stale information can be removed (flushed) and new information can be learned based on the new topology.

A *Topology Change* notification is sent whenever STP moves a port from the blocking state to the forwarding state. When it is received, the receiving device flushes the MAC learning entries for all ports that are not blocked other than the one where the notification was received, and also sends its own topology change notification out of those ports. In this way, it is guaranteed that stale information is removed from all the devices in the network.

Variants of STP

There are many variants of the Spanning Tree Protocol:

- Legacy STP (STP)—The original STP protocol was defined in IEEE 802.1D-1998. This creates a single spanning tree which is used for all VLANs and most of the convergence is timer-based.
- Rapid STP (RSTP)—This is an enhancement defined in IEEE 802.1D-2004 to provide more event-based, and hence faster, convergence. However, it still creates a single spanning tree for all VLANs.
- Multiple STP (MSTP)—A further enhancement was defined in IEEE 802.1Q-2005. This allows multiple spanning tree instances to be created over the same physical topology. By assigning different VLANs to the different spanning tree instances, data traffic can be load-balanced over different physical links. The number of different spanning tree instances that can be created is restricted to a much smaller number than the number of possible VLANs; however, multiple VLANs can be assigned to the same spanning tree instance. The BPDUs used to exchange MSTP information are always sent untagged; the VLAN and spanning tree instance data is encoded inside the BPDU.
- Per-Vlan STP (PVST)—This is an alternative mechanism for creating multiple spanning trees; it was developed by Cisco before the standardization of MSTP. Using PVST, a separate spanning tree is created for each VLAN. There are two variants: PVST+ (based on legacy STP), and PVRST (based on RSTP). At a packet level, the separation of the spanning trees is achieved by sending standard STP or RSTP BPDUs, tagged with the appropriate VLAN tag.
- Per-Vlan Rapid Spanning Tree (PVRST)— This feature is the IEEE 802.1w (RSTP) standard implemented per VLAN, and is also known as Rapid PVST or PVST+. A single instance of STP runs on each configured VLAN (if you do not manually disable STP). Each Rapid PVST+ instance on a VLAN has a single root switch. You can enable and disable STP on a per-VLAN basis when you are running Rapid PVST+.
- PVRST uses point-to-point wiring to provide rapid convergence of the spanning tree. The spanning tree reconfiguration can occur in less than one second with PVRST (in contrast to 50 seconds with the default settings in the 802.1D STP).
- REP (Cisco-proprietary ring-redundancy protocol)— This is a Cisco-proprietary protocol for providing resiliency in rings. It is included for completeness, as it provides MSTP compatibility mode, using which, it interoperates with an MSTP peer.

Multiple Spanning Tree Protocol Overview

The Multiple Spanning Tree Protocol (MSTP) is an STP variant that allows multiple and independent spanning trees to be created over the same physical network. The parameters for each spanning tree can be configured separately, so as to cause a different network devices to be selected as the root bridge or different paths to be selected to form the loop-free topology. Consequently, a given physical interface can be blocked for some of the spanning trees and unblocked for others.

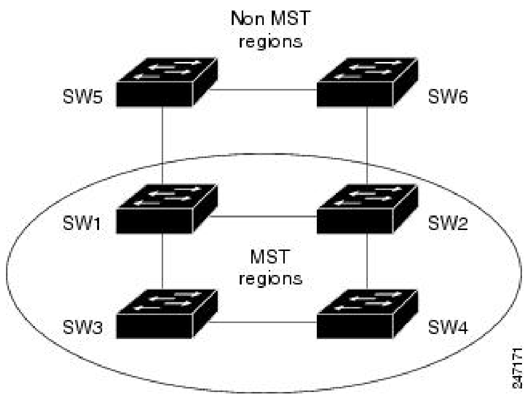
Having set up multiple spanning tree instances, the set of VLANs in use can be partitioned among them; for example, VLANs 1 - 100 can be assigned to spanning tree instance 1, VLANs 101 - 200 can be assigned to spanning tree instance 2, VLANs 201 - 300 can be assigned to spanning tree instance 3, and so on. Since each spanning tree has a different active topology with different active links, this has the effect of dividing the data traffic among the available redundant links based on the VLAN—a form of load balancing.

MSTP Regions

Along with supporting multiple spanning trees, MSTP also introduces the concept of regions. A region is a group of devices under the same administrative control and have similar configuration. In particular, the configuration for the region name, revision, and the mapping of VLANs to spanning tree instances must be identical on all the network devices in the region. A digest of this information is included in the BPDUs sent by each device, so as to allow other devices to verify whether they are in the same region.

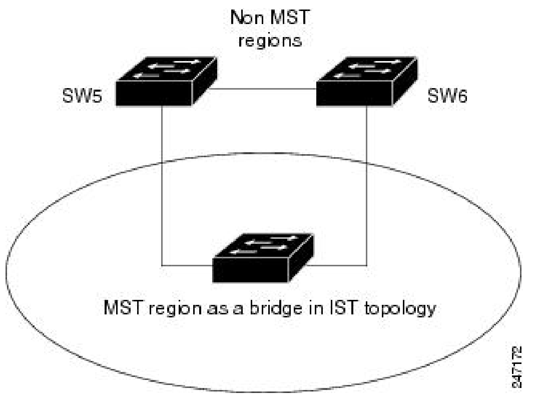
The following figure shows the operation of MST regions when bridges running MSTP are connected to bridges running legacy STP or RSTP. In this example, switches SW1, SW2, SW3, SW4 support MSTP, while switches SW5 and SW6 do not.

Figure 58: MST Interaction with Non-MST Regions



To handle this situation, an Internal Spanning Tree (IST) is used. This is always spanning tree instance 0 (zero). When communicating with non-MSTP-aware devices, the entire MSTP region is represented as a single switch. The logical IST topology in this case is shown in the following figure.

Figure 59: Logical Topology in MST Region Interacting with Non-MST Bridges



The same mechanism is used when communicating with MSTP devices in a different region. For example, SW5 in the above figure could represent a number of MSTP devices, all in a different region compared to SW1, SW2, SW3 and SW4.

MSTP Port Fast

MSTP includes a *Port Fast* feature for handling ports at the edge of the switched Ethernet network. For devices that only have one link to the switched network (typically host devices), there is no need to run MSTP, as there is only one available path. Furthermore, it is undesirable to trigger topology changes (and resultant MAC flushes) when the single link fails or is restored, as there is no alternative path.

By default, MSTP monitors ports where no BPDUs are received, and after a timeout, places them into *edge mode* whereby they do not participate in MSTP. However, this process can be speeded up (and convergence of the whole network thereby improved) by explicitly configuring edge ports as port fast.

Note

- You must disable and re-enable the port for Port Fast configuration to take effect. Use **shutdown** and **no shutdown** command (in interface configuration mode) to disable and re-enable the port.
- Port Fast is implemented as a Cisco-proprietary extension in Cisco implementations of legacy STP. However, it is encompassed in the standards for RSTP and MSTP, where it is known as Edge Port.

MSTP Root Guard

In networks with shared administrative control, it may be desirable for the network administrator to enforce aspects of the network topology and in particular, the location of the root bridge. By default, any device can become the root bridge for a spanning tree, if it has a lower priority or bridge ID. However, a more optimal forwarding topology can be achieved by placing the root bridge at a specific location in the centre of the network.

Note The administrator can set the root bridge priority to 0 in an effort to secure the root bridge position; however, this is no guarantee against another bridge which also has a priority of 0 and has a lower bridge ID.

The root guard feature provides a mechanism that allows the administrator to enforce the location of the root bridge. When root guard is configured on an interface, it prevents that interface from becoming a root port (that is, a port via which the root can be reached). If superior information is received via BPDUs on the interface that would normally cause it to become a root port, it instead becomes a backup or alternate port. In this case, it is placed in the blocking state and no data traffic is forwarded.

The root bridge itself has no root ports. Thus, by configuring root guard on every interface on a device, the administrator forces the device to become the root, and interfaces receiving conflicting information are blocked.

Note Root Guard is implemented as a Cisco-proprietary extension in Cisco implementations of legacy STP and RSTP. However, it is encompassed in the standard for MSTP, where it is known as Restricted Role.

MSTP Topology Change Guard

In certain situations, it may be desirable to prevent topology changes originating at or received at a given port from being propagated to the rest of the network. This may be the case, for example, when the network is not under a single administrative control and it is desirable to prevent devices external to the core of the network from causing MAC address flushing in the core. This behavior can be enabled by configuring Topology Change Guard on the port.

Note Topology Change Guard is known as *Restricted TCN* in the MSTP standard.

MSTP Supported Features

Cisco ASR 9000 Series Routers support MSTP, as defined in IEEE 802.1Q-2005, on physical Ethernet interfaces and Ethernet Bundle interfaces. Note that this includes the Port Fast, Backbone Fast, Uplink Fast and Root Guard features found in Cisco implementations of legacy STP, RSTP and PVST, as these are encompassed by the standard MSTP protocol. Cisco ASR 9000 Series Routers can operate in either standard 802.1Q mode, or in Provide Edge (802.1ad) mode. In provider edge mode, a different MAC address is used for BPDUs, and any BPDUs received with the 802.1Q MAC address are forwarded transparently.

In addition, these additional Cisco features are supported:

- BPDU Guard—This Cisco feature protects against misconfiguration of edge ports.
- Flush Containment—This Cisco feature helps prevent unnecessary MAC flushes that would otherwise occur following a topology change.
- Bringup Delay—This Cisco feature prevents an interface from being added to the active topology before it is ready to forward traffic.

Note Interoperation with RSTP is supported, as described in the 802.1Q standard; however, interoperation with legacy STP is not supported.

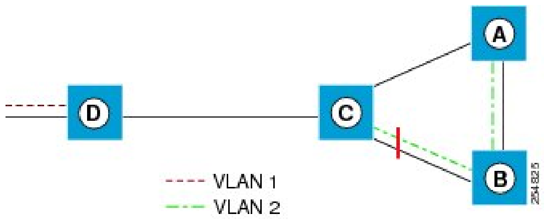
BPDU Guard

BPDU Guard is a Cisco feature that protects against misconfiguration of edge ports. It is an enhancement to the MSTP port fast feature. When port fast is configured on an interface, MSTP considers that interface to be an edge port and removes it from consideration when calculating the spanning tree. When BPDU Guard is configured, MSTP additionally shuts down the interface using error-disable if an MSTP BPDU is received.

Flush Containment

Flush containment is a Cisco feature that helps prevent unnecessary MAC flushes due to unrelated topology changes in other areas of a network. This is best illustrated by example. The following figure shows a network containing four devices. Two VLANs are in use: VLAN 1 is only used on device D, while VLAN 2 spans devices A, B and C. The two VLANs are in the same spanning tree instance, but do not share any links.

Figure 60: Flush Containment



If the link AB goes down, then in normal operation, as C brings up its blocked port, it sends out a topology change notification on all other interfaces, including towards D. This causes a MAC flush to occur for VLAN 1, even though the topology change which has taken place only affects VLAN 2.

Flush containment helps deal with this problem by preventing topology change notifications from being sent on interfaces on which no VLANs are configured for the MSTI in question. In the example network this would mean no topology change notifications would be sent from C to D, and the MAC flushes which take place would be confined to the right hand side of the network.

Note Flush containment is enabled by default, but can be disabled by configuration, thus restoring the behavior described in the IEEE 802.1Q standard.

Bringup Delay

Bringup delay is a Cisco feature that stops MSTP from considering an interface when calculating the spanning tree, if the interface is not yet ready to forward traffic. This is useful when a line card first boots up, as the system may declare that the interfaces on that card are *Up* before the dataplane is fully ready to forward traffic. According to the standard, MSTP considers the interfaces as soon as they are declared *Up*, and this may cause it to move other interfaces into the blocking state if the new interfaces are selected instead.

Bringup delay solves this problem by adding a configurable delay period which occurs as interfaces that are configured with MSTP first come into existence. Until this delay period ends, the interfaces remain in blocking state, and are not considered when calculating the spanning tree.

Bringup delay only takes place when interfaces which are already configured with MSTP are created, for example, on a card reload. No delay takes place if an interface which already exists is later configured with MSTP.

Restrictions for configuring MSTP

These restrictions apply when using MSTP:

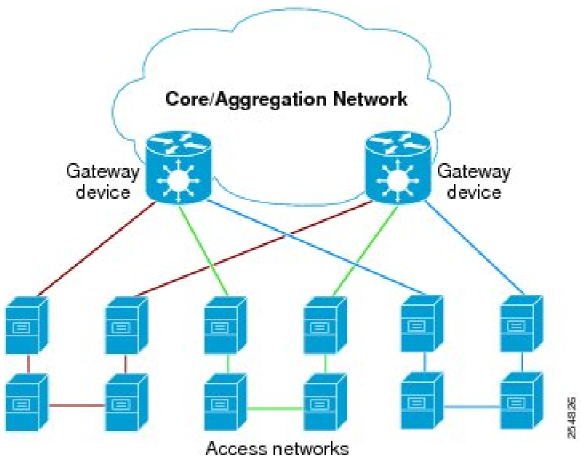
- MSTP must only be enabled on interfaces where the interface itself (if it is in L2 mode) or all of the subinterfaces have a simple encapsulation configured. These encapsulation matching criteria are considered simple:
    - Single-tagged 802.1Q frames
    - Double-tagged Q-in-Q frames (only the outermost tag is examined)
    - 802.1ad frames (if MSTP is operating in Provider Bridge mode)
    - Ranges or lists of tags (any of the above)
- If an L2 interface or subinterface is configured with an encapsulation that matches multiple VLANs, then all of those VLANs must be mapped to the same spanning tree instance. There is therefore a single spanning tree instance associated with each L2 interface or subinterface.
- All the interfaces or subinterfaces in a given bridge domain must be associated with the same spanning tree instance.
- Multiple subinterfaces on the same interface must not be associated with the same spanning tree instance, unless those subinterfaces are in the same split horizon group. In other words, hair-pinning is not possible.
- Across the network, L2 interfaces or subinterfaces must be configured on all redundant paths for all the VLANs mapped to each spanning tree instance. This is to avoid inadvertent loss of connectivity due to STP blocking of a port.

Caution A subinterface with a default or untagged encapsulation will lead to an MSTP state machine failure.

Access Gateway

One common deployment scenario for router is as an nPE gateway device situated between a network of uPE access devices and a core or aggregation network. Each gateway device may provide connectivity for many access networks, as shown in following figure. The access networks (typically rings) have redundant links to the core or aggregation network, and therefore must use some variant of STP or a similar protocol to ensure the network remains loopfree.

Figure 61: Core or Aggregation Network



It is possible for the gateway devices to also participate in the STP protocol. However, since each gateway device may be connected to many access networks, this would result in one of two solutions:

- A single topology is maintained covering all of the access networks. This is undesirable as it means topology changes in one access network could impact all the other access networks.
- The gateway devices runs multiple instances of the STP protocol, one for each access network. This means a separate protocol database and separate protocol state machines are maintained for each access network, which is undesirable due to the memory and CPU resource that would be required on the gateway device.

It can be seen that both of these options have significant disadvantages.

Another alternative is for the gateway devices to tunnel protocol BPDUs between the *legs* of each access network, but not to participate in the protocol themselves. While this results in correct loopfree topologies, it also has significant downsides:

- Since there is no direct connection between the *legs* of the access ring, a failure in one of the leg links is not immediately detected by the access device connected to the other *leg*. Therefore, recovery from the failure must wait for protocol timeouts, which leads to a traffic loss of at least six seconds.
- As the gateway devices do not participate in the protocol, they are not aware of any topology changes in the access network. The aggregation network may therefore direct traffic destined for the access network over the wrong *leg*, following a topology change. This can lead to traffic loss on the order of the MAC learning timeout (5 minutes by default).

Access gateway is a Cisco feature intended to address this deployment scenario, without incurring the disadvantages of the solutions described above.

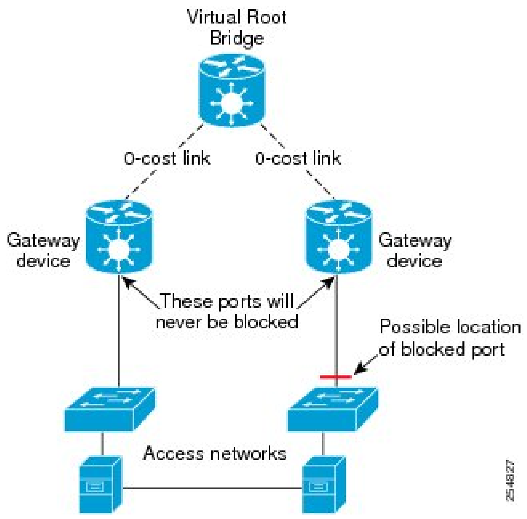
Overview of Access Gateway

Access gateway is based on two assumptions:

- Both gateway devices provide connectivity to the core or aggregation network at all times. Generally, resiliency mechanisms used within the core or aggregation network are sufficient to ensure this is the case. In many deployments, VPLS is used in the core or aggregation network to provide this resiliency.
- The desired root of all of the spanning trees for each access network is one of the gateway devices. This will be the case if (as is typical) the majority of the traffic is between an access device and the core or aggregation network, and there is little if any traffic between the access devices.

With these assumptions, an STP topology can be envisaged where for every spanning tree, there is a virtual root bridge behind (that is, on the core side of) the gateway devices, and both gateway devices have a zero cost path to the virtual root bridge. In this case, the ports that connect the gateway devices to the access network would never be blocked by the spanning tree protocol, but would always be in the forwarding state. This is illustrated in the following figure.

Figure 62: Access Network



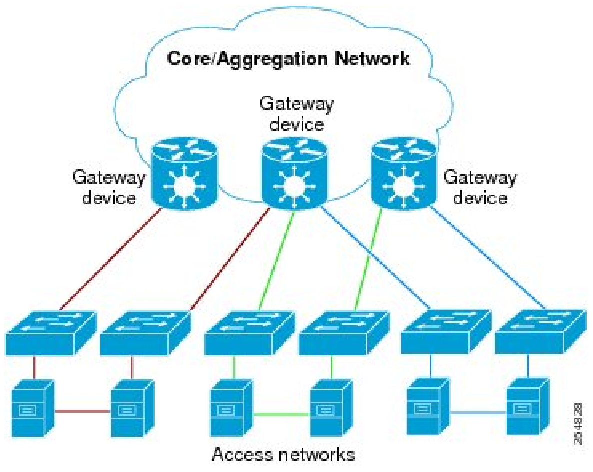
With this topology, it can be observed that the BPDUs sent by the gateway devices are constant: since the root bridge never changes (as we assume the aggregation or core network always provides connectivity) and the ports are always forwarding, the information sent in the BPDUs never changes.

Access gateway makes use of this by removing the need to run the full STP protocol and associated state machines on the gateway devices, and instead just sends statically configured BPDUs towards the access network. The BPDUs are configured so as to mimic the behavior above, so that they contain the same information that would be sent if the full protocol was running. To the access devices, it appears that the gateway devices are fully participating in the protocol; however, since in fact the gateway devices are just sending static BPDUs, very little memory or CPU resource is needed on the gateway devices, and many access networks can be supported simultaneously.

For the most part, the gateway devices can ignore any BPDUs received from the access network; however, one exception is when the access network signals a topology change. The gateway devices can act on this appropriately, for example by triggering an LDP MAC withdrawal in the case where the core or aggregation network uses VPLS.

In many cases, it is not necessary to have direct connectivity between the gateway devices; since the gateway devices statically send configured BPDUs over the access links, they can each be configured independently (so long as the configuration on each is consistent). This also means that different access networks can use different pairs of gateway devices, as shown in the following figure.

Figure 63: Access Networks



Note Although the above figure shows access rings, in general there are no restrictions on the access network topology or the number or location of links to the gateway devices.

Access gateway ensures loop-free connectivity in the event of these failure cases:

- Failure of a link in the access network.
- Failure of a link between the access network and the gateway device.
- Failure of an access device.
- Failure of a gateway device.

Topology Change Propagation

There is one case where the two gateway devices need to exchange BPDUs between each other, and this is to handle topology changes in the access network. If a failure in the access network results in a topology change that causes a previously blocked port to move to forwarding, the access device sends a topology change notification out on that port, so as to notify the rest of the network about the change and trigger the necessary MAC learning flushes. Typically, the topology change notification is sent towards the root bridge, in the case of access gateway, that means it is sent to one of the gateway devices.

As described above, this causes the gateway device itself to take any necessary action; however, if the failure caused the access network to become partitioned, it may also be necessary to propagate the topology change notification to the rest of the access network, that is, the portion connected to the other gateway device. This can be achieved by ensuring there is connectivity between the gateway devices, so that each gateway device can propagate any topology change notifications it receives from the access network to the other device. When a gateway device receives a BPDU from the other gateway device that indicates a topology change, it signals this in the static BPDUs (that it is sending towards the access network).

Topology Change Propagation is only necessary when these two conditions are met:

- The access network contains three or more access devices. If there are fewer than three devices, then any possible failure must be detected by all the devices.
- The access devices send traffic to each other, and not just to or from the core or aggregation network. If all the traffic is to or from the core or aggregation network, then all the access devices must either already

be sending traffic in the right direction, or will learn about the topology change from the access device that originates it.

Preempt Delay

One of the assumptions underpinning access gateway is that the gateway devices are always available to provide connectivity to the core or aggregation network. However, there is one situation where this assumption may not hold, which is at bringup time. At bringup, it may be the case that the access facing interface is available before all of the necessary signaling and convergence has completed that means traffic can successfully be forwarded into the core or aggregation network. Since access gateway starts sending BPDUs as soon as the interface comes up, this could result in the access devices sending traffic to the gateway device before it is ready to receive it. To avoid this problem, the preempt delay feature is used.

The preempt delay feature causes access gateway to send out inferior BPDUs for some period of time after the interface comes up, before reverting to the normal values. These inferior BPDUs can be configured such that the access network directs all traffic to the other gateway device, unless the other gateway device is also down. If the other gateway device is unavailable, it is desirable for the traffic to be sent to this device, even if it is only partially available, rather than being dropped completely. For this reason, inferior BPDUs are sent during the preempt delay time, rather than sending no BPDUs at all.

Supported Access Gateway Protocols

Access Gateway is supported on routers when the following protocols are used in the access network

Table 7: Protocols

Access Network Protocol	Access Gateway Variant
MSTP	MST Access Gateway (MSTAG)
REP	REP Access gateway (REPAG)[1]
PVST+	PVST+ Access Gateway (PVSTAG) [2]
PVRST	PVRST Access Gateway (PVRSTAG) [3]

1. REP Access Gateway is supported when the access device interfaces that connect to the gateway devices are configured with REP MSTP Compatibility mode.
2. Topology Change Propagation is not supported for PVSTAG.
3. Topology Change Propagation is not supported for PVRSTAG.

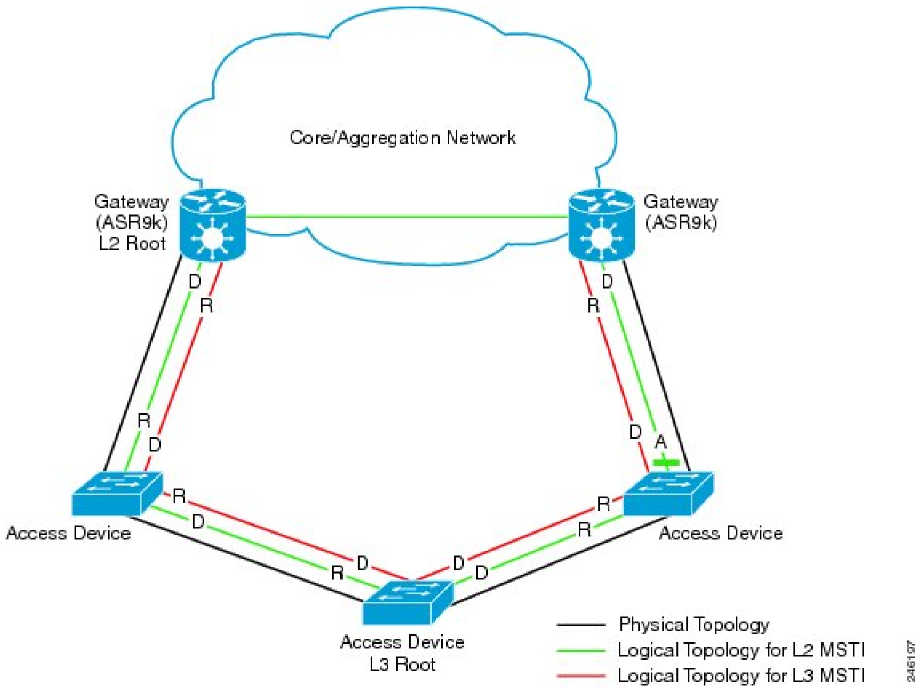
MSTAG Edge Mode

An access gateway is used in a Layer 2 (L2) environment to ensure that for each Multiple Spanning Tree Instance (MSTI), each access device has one path to the core or aggregation network. The core or aggregation network provides L2 (Ethernet) connectivity between two gateway devices. Therefore, when there are no failures, there must be at least one blocked port in the access network for each MSTI. In the case of an access ring, there should be one blocked port in the access ring. For each MSTI – this is typically one of the uplink ports that connects to one of the gateway devices. This is achieved by configuring MSTAG in such a way that the gateway devices appear to have the best path to the best possible Multiple Spanning Tree Protocol (MSTP) root node. Thus, the access devices always use the gateway devices to reach the root, and the ports on the gateway devices are always in the designated forwarding state.

In a mixed Layer 2-Layer 3 environment, the L2 access network is used to provide a Layer 2 service on certain VLANs and a Layer 3 (L3) service on other VLANs. In the access network, a different MSTI is used for the L2 service and the L3 service. For the L2 VLANs, the core or aggregation network provides L2 connectivity between the gateway devices. However, for the L3 service, the gateway devices terminate the L2 network and perform L3 routing. Typically, an L3 redundancy mechanism such as HSRP or VRRP is used to allow the end hosts to route to the correct gateway.

In this scenario, the use of MSTAG alone does not achieve the desired behavior for the L3 MSTI. This is because it results in one of the ports in the access network being blocked, even though there is actually no loop. (This, in turn, is because there is no L2 connectivity between the gateway devices for the L3 VLANs.) In fact, because the gateway devices terminate the L2 network for the L3 VLANs, the desirable behavior is for the MSTP root to be located in the access network, and for the gateway devices to appear as leaf nodes with a single connection. This can be achieved by reversing the MSTAG configuration; that is, setting the gateway devices to advertise the worst possible path to the worst possible root. This forces the access devices to elect one of the access devices as the root, and therefore, no ports are blocked. In this case, the ports on the gateway devices are always in root forwarding state. The MSTAG Edge mode feature enables this scenario by changing the role advertised by the gateway devices from designated to root. The following figure illustrates this scenario.

Figure 64: MSTAG Edge Mode scenario



- D - Designated port (forwarding)
- R - Root port (forwarding)
- A - Alternate port (blocked)

For normal MSTAG, and for the L2 MSTIs, topology change notifications are propagated from one gateway device to the other, and re-advertised into the access network. However, for the L3 MSTI, this is not desirable. As there is no block for the L3 MSTI in the access network, the topology change notification could loop forever. To avoid that situation, MSTAG Edge mode completely disables handling of topology change notifications in the gateway devices.

PVSTAG on Bundle Interfaces

Per-VLAN Spanning Tree Access Gateway (PVSTAG) support has been extended on bundle interfaces, along with physical interfaces, to cater to an increasing number of customers that support PVST access networks.

For physical interfaces, bridge protocol data units (BDPUs) are sent from the line card that hosts the interfaces. However, for bundle interfaces BPDUs are sent from the route processor (RP). When an RP failover occurs, the data traffic flowing over the bundle interface is not affected; as a result, no BPDUs are sent until the failover is complete and the newly active RP takes over. If there is a delay, the peer device times out the BPDU information. This leads to a forwarding loop, which results in disruption in Ethernet networks. It is therefore important to ensure that, upon RP failover, the peer device does not time out the BPDU information.

Per-VLAN Rapid Spanning Tree

Per-VLAN Rapid Spanning Tree (PVRST) or Rapid PVST or PVST+ is the IEEE 802.1w (RSTP) standard implemented per VLAN. A single instance of STP runs on each configured VLAN (if you do not manually disable STP). Each Rapid PVST+ instance on a VLAN has a single root switch. You can enable and disable STP on a per-VLAN basis when you are running Rapid PVST+.

PVRST uses point-to-point wiring to provide rapid convergence of the spanning tree. The spanning tree reconfiguration can occur in less than 1 second with PVRST (in contrast to 50 seconds with the default settings in the 802.1D STP).

Note PVRST supports one STP instance for each VLAN.

Using PVRST, STP convergence occurs rapidly. Each designated or root port in the STP sends out a BPDU every 2 seconds by default. On a designated or root port in the topology, if hello messages are missed three consecutive times, or if the maximum age expires, the port immediately flushes all protocol information in the table. A port considers that it loses connectivity to its direct neighbor root or designated port if it misses three BPDUs or if the maximum age expires. This rapid aging of the protocol information allows quick failure detection.

PVRST achieves rapid transition to the forwarding state only on edge ports and point-to-point links. Although the link type is configurable, the system automatically derives the link type information from the duplex setting of the port. Full-duplex ports are assumed to be point-to-point ports, while half-duplex ports are assumed to be shared ports.

Disadvantages

- Increased load in terms of increased packet rate.
- Greater CPU and memory utilization due to one STP instance for every LAN.

Implementation of PVRST in IOS-XR

The implementation of PVRST in IOS-XR has the following characteristics:

- Configuration of the Forward Delay and Max Age timers is only supported globally and not per VLAN.
- Configuration of the Hello timer is supported per port and not per VLAN. The Hello timer configured on a port applies to all VLANs on that specific port.
- The cost of a spanning tree bundle port is always 10000. It is not affected by any of the following:

  - Number or speed of the bundle members
  - Logical or administrative operational status of the bundle member ports
  - Addition or deletion of bundle members

- Receiving BPDU on an interface configured with the BPDU Guard error-disables the physical interface as well as any layer-2 or layer-3 sub-interfaces configured on the physical interface.
- Only Ethernet Flow-points (EFPs) that are untagged or have a single VLAN tag can be protected by PVRST.
- If any one EFP in a bridge-domain is protected by PVRST, then all EFPs in that bridge domain must belong to the same VLAN.
- If any one EFP on a port is protected by PVRST, then all EFPs on that port must be protected by PVRST.
- PVRST supports up to 8000 Port Tree Instances (PTIs) per line card and 16000 PTIs per system. PTI refers to the product of the number of ports and VLANs. PVRST supports 1000 VLANs.

Multiple VLAN Registration Protocol

The Multiple VLAN Registration Protocol is defined in IEEE 802.1ak and is used in MSTP based networks to optimize the propagation of multicast and broadcast frames.

By default, multicast and broadcast frames are propagated to every point in the network, according to the spanning tree, and hence to every edge (host) device that is attached to the network. However, for a given VLAN, it may be the case that only certain hosts are interested in receiving the traffic for that VLAN. Furthermore, it may be the case that a given network device, or even an entire segment of the network, has no attached hosts that are interested in receiving traffic for that VLAN. In this case, an optimization is possible by avoiding propagating traffic for that VLAN to those devices that have no stake in it. MVRP provides the necessary protocol signaling that allows each host and device to indicate to its attached peers which VLANs it is interested in.

MVRP-enabled devices can operate in two modes:

- Static mode—In this mode, the device initiates MVRP messages declaring interest in a statically configured set of VLANs. Note that the protocol is still dynamic with respect to the MSTP topology; it is the set of VLANs that is static.
- Dynamic mode—In this mode, the device processes MVRP messages received on different ports, and aggregates them dynamically to determine the set of VLANs it is interested in. It sends MVRP messages declaring interest in this set. In dynamic mode, the device also uses the received MVRP messages to prune the traffic sent out of each port so that traffic is only sent for the VLANs that the attached device has indicated it is interested in.

The router supports operating in static mode. This is known as MVRP-lite.

How to Implement Multiple Spanning Tree Protocol

This section contains these procedures:

Configuring MSTP

This section describes the procedure for configuring MSTP:

Note This section does not describe how to configure data switching. Refer to the Implementing Multipoint Layer 2 Services module for more information.

Enabling MSTP

By default, STP is disabled on all interfaces. MSTP should be explicitly enabled by configuration on each physical or Ethernet Bundle interface. When MSTP is configured on an interface, all the subinterfaces of that interface are automatically MSTP-enabled.

Configuring MSTP parameters

The MSTP Standard defines a number of configurable parameters. The global parameters are:

- Region Name and Revision
- Bringup Delay
- Forward Delay
- Max Age or Hops
- Transmit Hold Count
- Provider Bridge mode
- Flush Containment
- VLAN IDs (per spanning-tree instance)
- Bridge Priority (per spanning-tree instance)

The per-interface parameters are:

- External port path cost
- Hello Time
- Link Type
- Port Fast and BPDU Guard
- Root Guard and Topology Change Guard
- Port priority (per spanning-tree instance)
- Internal port path cost (per spanning-tree instance)

Per-interface configuration takes place in an interface submode within the MST configuration submode.

Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default value.

SUMMARY STEPS

1. **configure**
2. **spanning-tree mst** *protocol instance identifier*
3. **bringup delay for** *interval* { **minutes** | **seconds** }
4. **flush containment disable**
5. **name** *name*
6. **revision** *revision* -number
7. **forward-delay** *seconds*
8. **maximum** { **age** *seconds* | **hops** *hops* }
9. **transmit hold-count** *count*
10. **provider-bridge**
11. **instance** *id*
12. **priority** *priority*
13. **vlan-id** *vlan-range* [,*vlan-range*][,*vlan-range*] [,*vlan-range*]
14. **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance*
15. **instance** *id* **port-priority** *priority*
16. **instance** *id* **cost** *cost*
17. **external-cost** *cost*
18. **link-type** { **point-to-point** | **multipoint** }
19. **hello-time** *seconds*
20. **portfast** [**bpdu-guard**]
21. **guard root**
22. **guard topology-change**
23. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **spanning-tree mst** *protocol instance identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# spanning-tree mst a
RP/0/RSP0/CPU0:router(config-mstp)#
```

Enters the MSTP configuration submode.

Step 3 **bringup delay for** *interval* { **minutes** | **seconds** }

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)#bringup delay for 10 minutes
```

Configures the time interval to delay bringup for.

Step 4 **flush containment disable**

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)#flush containment disable
```

Disable flush containment.

This command performs MAC flush on all instances regardless of the their state.

Step 5 **name** *name*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# name m1
```

Sets the name of the MSTP region.

The default value is the MAC address of the switch, formatted as a text string by means of the hexadecimal representation specified in IEEE Std 802.

Step 6 **revision** *revision* -number

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# revision 10
```

Sets the revision level of the MSTP region.

Allowed values are from 0 through 65535.

Step 7 **forward-delay** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# forward-delay 20
```

Sets the forward-delay parameter for the bridge.

Allowed values for bridge forward-delay time in seconds are from 4 through 30.

Step 8 **maximum** { **age** *seconds* | **hops** *hops* }

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# max age 40
RP/0/RSP0/CPU0:router(config-mstp)# max hops 30
```

Sets the maximum age and maximum hops performance parameters for the bridge.

Allowed values for maximum age time for the bridge in seconds are from 6 through 40.

Allowed values for maximum number of hops for the bridge in seconds are from 6 through 40.

Step 9 **transmit hold-count** *count*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# transmit hold-count 8
```

Sets the transmit hold count performance parameter.

Allowed values are from 1 through 10.

Step 10 **provider-bridge**

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# provider-bridge
```

Places the current instance of the protocol in 802.1ad mode.

Step 11 **instance** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# instance 101
RP/0/RSP0/CPU0:router(config-mstp-inst)#
```

Enters the MSTI configuration submode.

Allowed values for the MSTI ID are from 0 through 4094.

Step 12 **priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-inst)# priority 8192
```

Sets the bridge priority for the current MSTI.

Allowed values are from 0 through 61440 in multiples of 4096.

Step 13 **vlan-id** *vlan-range* [,*vlan-range*][,*vlan-range*] [,*vlan-range*]

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-inst)# vlan-id 2-1005
```

Associates a set of VLAN IDs with the current MSTI.

List of VLAN ranges in the form a-b, c, d, e-f, g, and so on.

Note Repeat steps 11 to 13 for each MSTI.

Step 14 **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)# interface FastEthernet 0/0/0/1
RP/0/RSP0/CPU0:router(config-mstp-if)#
```

Enters the MSTP interface configuration submode, and enables STP for the specified port.

Forward interface in Rack/Slot/Instance/Port format.

Step 15 **instance** *id* **port-priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# instance 101 port-priority 160
```

Sets the port priority performance parameter for the MSTI.

Allowed values for the MSTI ID are from 0 through 4094.

Allowed values for port priority are from 0 through 240 in multiples of 16.

Step 16 **instance** *id* **cost** *cost*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# instance 101 cost 10000
```

Sets the internal path cost for a given instance on the current port.

Allowed values for the MSTI ID are from 0 through 4094.

Allowed values for port cost are from 1 through 200000000.

Repeat steps 15 and 16 for each MSTI for each interface.

Step 17 **external-cost** *cost*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# external-cost 10000
```

Sets the external path cost on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 18 **link-type** { **point-to-point** | **multipoint** }

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# link-type point-to-point
```

Sets the link type of the port to point-to-point or multipoint.

Step 19 **hello-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are 1 and 2.

Step 20 **portfast** [**bpdu-guard**]

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# portfast
RP/0/RSP0/CPU0:router(config-mstp-if)# portfast bpduguard
```

Enables PortFast on the port, and optionally enables BPDU guard.

Step 21 **guard root**

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# guard root
```

Enables RootGuard on the port.

Step 22 **guard topology-change**

Example:

```
RP/0/RSP0/CPU0:router(config-mstp-if)# guard topology-change
```

Enables TopologyChangeGuard on the port.

Note Repeat steps 14 to 22 for each interface.

Step 23 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Verifying MSTP

These show commands allow you to verify the operation of MSTP:

- **show spanning-tree mst** *mst-name*
- **show spanning-tree mst** *mst-name* **interface** *interface-name*
- **show spanning-tree mst** *mst-name* **errors**
- **show spanning-tree mst** *mst-name* **configuration**
- **show spanning-tree mst** *mst-name* **bpdu interface** *interface-name*

- **show spanning-tree mst** *mst-name* **topology-change flushes**

Configuring MSTAG or REPAG

This section describes the procedures for configuring MSTAG:

Note The procedures for configuring REPAG are identical.

This section does not describe how to configure data switching. Refer to the Implementing Multipoint Layer 2 Services module for more information.

Configuring an untagged subinterface

In order to enable MSTAG on a physical or Bundle Ethernet interface, an L2 subinterface must first be configured which matches untagged packets, using the encapsulation untagged command. Refer to *The Cisco ASR 9000 Series Routers Carrier Ethernet Model* module for more information about configuring L2 subinterfaces.

Enabling MSTAG

MSTAG is enabled on a physical or Bundle Ethernet interface by explicitly configuring it on the corresponding untagged subinterface. When MSTAG is configured on the untagged subinterface, it is automatically enabled on the physical or Bundle Ethernet interface and on all other subinterfaces on that physical or Bundle Ethernet subinterface.

Configuring MSTAG parameters

MSTAG parameters are configured separately on each interface, and MSTAG runs completely independently on each interface. There is no interaction between the MSTAG parameters on different interfaces (unless they are connected to the same access network).

These parameters are configurable for each interface:

- Region Name and Revision
- Bridge ID
- Port ID
- External port path cost
- Max Age
- Provide Bridge mode
- Hello Time

The following MSTAG parameters are configurable for each interface, for each spanning tree instance:

- VLAN IDs
- Root Bridge Priority and ID

- Bridge Priority
- Port Priority
- Internal Port Path Cost

To ensure consistent operation across the access network, these guidelines should be used when configuring:

- Both gateway devices should be configured with a Root Bridge Priority and ID (for each spanning tree instance) that is better (lower) than the Bridge Priority and Bridge ID of any device in the access network. It is recommended to set the Root Bridge Priority and ID to 0 on the gateway devices.

Note To avoid an STP dispute being detected by the access devices, the same root priority and ID should be configured on both gateway devices.

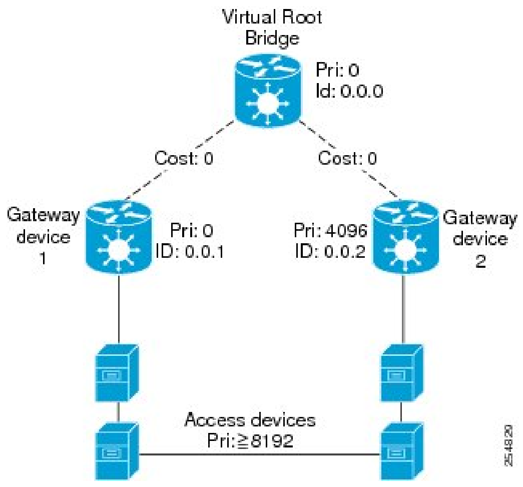
- Both gateway devices should be configured with a Port Path Cost of 0.
- For each spanning tree instance, one gateway device should be configured with the bridge priority and ID that is higher than the root bridge priority and ID, but lower than the bridge priority and ID of any other device in the network (including the other gateway device). It is recommended to set the bridge priority to 0.
- For each spanning tree instance, the second gateway device should be configured with a bridge priority and ID that is higher than the root bridge priority and ID and the first gateway device bridge priority and ID, but lower than the bridge priority and ID of any device in the access network. It is recommended to set the bridge priority to 4096 (this is the lowest allowable value greater than 0).
- All of the access devices should be configured with a higher bridge priority than the gateway devices. It is recommended to use values of 8192 or higher.
- For each spanning tree instance, the port path cost and other parameters may be configured on the access devices so as to ensure the desired port is put into the blocked state when all links are up.

Caution There are no checks on MSTAG configuration—misconfiguration may result in incorrect operation of the MSTP protocol in the access devices (for example, an STP dispute being detected).

The guidelines above are illustrated in the following figure.

Note These guidelines do not apply to REPAG, as in that case the access devices ignore the information received from the gateway devices apart from when a topology change is signalled.

Figure 65: MSTAG Guidelines



Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default values.

SUMMARY STEPS

1. **configure**
2. **spanning-tree mstag** *protocol instance identifier*
3. **preempt delay for** interval{ **seconds** | **minutes** | **hours** }
4. **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance.subinterface*
5. **name** *name*
6. **revision** *revision* -number
7. **max age** *seconds*
8. **provider-bridge**
9. **bridge-id** *id*
10. **port-id** *id*
11. **external-cost** *cost*
12. **hello-time** *seconds*
13. **instance** *id*
14. **edge mode**
15. **vlan-id** *vlan-range* [, *vlan-range*] [,*vlan-range*] [,*vlan-range*]
16. **priority** *priority*
17. **port-priority** *priority*
18. **cost** *cost*
19. **root-bridge** *id*
20. **root-priority** *priority*
21. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **spanning-tree mstag** *protocol instance identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# spanning-tree mstag a
RP/0/RSP0/CPU0:router(config-mstag)#
```

Enters the MSTAG configuration submode.

Step 3 **preempt delay for** interval{ **seconds** | **minutes** | **hours** }

Example:

```
RP/0/RSP0/CPU0:router(config-mstag)# preempt delay for 10 seconds
```

Specifies the delay period during which startup BPDUs should be sent, before preempting.

Step 4 **interface** { **Bundle-Ether** | **GigabitEthernet** | **TenGigE** | **FastEthernet** } *instance.subinterface*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag)# interface GigabitEthernet0/2/0/30.1
RP/0/RSP0/CPU0:router(config-mstag-if)#
```

Enters the MSTAG interface configuration submode, and enables MSTAG for the specified port.

Step 5 **name** *name*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# name leo
```

Sets the name of the MSTP region.

The default value is the MAC address of the switch, formatted as a text string using the hexadecimal representation specified in IEEE Standard 802.

Step 6 **revision** *revision* -number

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# revision 1
```

Sets the revision level of the MSTP region.

Allowed values are from 0 through 65535.

Step 7 **max age** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# max age 20
```

Sets the maximum age performance parameters for the bridge.

Allowed values for the maximum age time for the bridge in seconds are from 6 through 40.

Step 8 **provider-bridge**

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# provider-bridge
```

Places the current instance of the protocol in 802.1ad mode.

Step 9 **bridge-id** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# bridge-id 001c.0000.0011
```

Sets the bridge ID for the current switch.

Step 10 **port-id** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# port-id 111
```

Sets the port ID for the current switch.

Step 11 **external-cost** *cost*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# external-cost 10000
```

Sets the external path cost on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 12 **hello-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are from 1 through 2.

Step 13 **instance** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if)# instance 1
```

Enters the MSTI configuration submode.

Allowed values for the MSTI ID are from 0 through 4094.

Step 14 **edge mode**

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# edge mode
```

Enables access gateway edge mode for this MSTI.

Step 15 **vlan-id** *vlan-range* [, *vlan-range*] [,*vlan-range*] [,*vlan-range*]

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# vlan-id 2-1005
```

Associates a set of VLAN IDs with the current MSTI.

List of VLAN ranges in the form a-b, c, d, e-f, g, and so on.

Step 16 **priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# priority 4096
```

Sets the bridge priority for the current MSTI.

Allowed values are from 0 through 61440 in multiples of 4096.

Step 17 **port-priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# port-priority 160
```

Sets the port priority performance parameter for the MSTI.

Allowed values for port priority are from 0 through 240 in multiples of 16.

Step 18 **cost** *cost*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# cost 10000
```

Sets the internal path cost for a given instance on the current port.

Allowed values for port cost are from 1 through 200000000.

Step 19 **root-bridge** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# root-id 001c.0000.0011
```

Sets the root bridge ID for the BPDUs sent from the current port.

Step 20 **root-priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-mstag-if-inst)# root-priority 4096
```

Sets the root bridge priority for the BPDUs sent from this port.

Note Repeat steps 4 to 19 to configure each interface, and repeat steps 13 to 19 to configure each MSTI for each interface.

Step 21 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MSTAG Topology Change Propagation

MSTAG Topology Change Propagation is configured simply by configuring connectivity between the MSTAG-enabled interfaces on the two gateway devices:

1. Configure MSTAG as described above. Take note of the untagged subinterface that is used.
2. Configure connectivity between the gateway devices. This may be via an MPLS Pseudowire, or may be a VLAN subinterface if there is a direct physical link.
3. Configure a point-to-point (P2P) cross-connect on each gateway device that contains the untagged subinterface and the link (PW or subinterface) to the other gateway device.

Once the untagged subinterface that is configured for MSTAG is added to the P2P cross-connect, MSTAG Topology Change Propagation is automatically enabled. MSTAG forwards BDPUs via the cross-connect to the other gateway device, so as to signal when a topology change has been detected.

For more information on configuring MPLS pseudowire or P2P cross-connects, refer to the Implementing Point to Point Layer 2 Services module.

Verifying MSTAG

These show commands allow you to verify the operation of MSTAG:

- **show spanning-tree mstag** *mst-name*

- **show spanning-tree mstag** *mst-name* **bpdu interface** *interface-name*
- **show spanning-tree mstag** *mst-name* **topology-change flushes**

Analogous commands are available for REPAG.

MSTAG Uplink Tracking

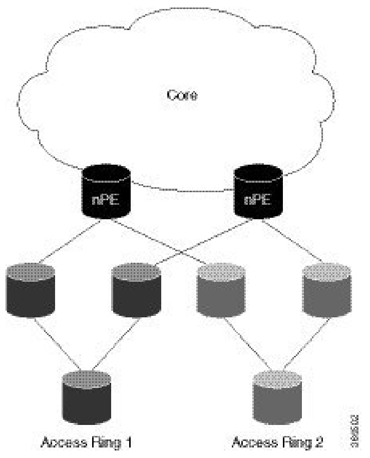
The MSTAG Uplink Tracking feature monitors the connectivity of an nPE gateway device to the core or aggregation network. This feature prevents traffic loss if there is a connectivity failure between a gateway router and the core network. This feature also ensures reduction in the frequency of traffic outages and reduced need for redundancy in connectivity from the gateway to the core network.

Multiple Spanning Tree Access Gateway (MSTAG) tracks the connectivity of interfaces that face the core. When an nPE device loses connectivity to the core, it sends start-up BPDUs indicating that all core-facing interfaces are down. This allows the access ring to switch the traffic to go through the other nPE device.

The core connectivity is based on a per-protocol instance. This is because each access ring corresponds to an access ring, and they may use different interfaces to forward traffic to the core. Therefore, it is possible for different protocol instances to have different core connectivity status.

In this topology there are two access rings. Each one is connected to the core through a pair of nPE devices in which MSTAG is configured. MSTAG allows each access ring to run the STP independently. When one of the nPE devices lose core connectivity, it starts sending start-up BPDUs indicating that traffic must flow through the other side of the access ring. Once core connectivity is available, the nPE device starts sending the standard BPDUs. If pre-empt delay is set, it continues to send the start-up BPDUs until the timer expires. For example, if pre-empt delay is configured as ten seconds, the nPE device sends startup BPDUs for the first ten seconds after core connectivity becomes available. After ten seconds, it starts sending standard BPDUs.

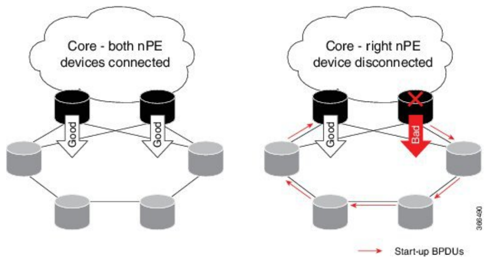
Figure 66: MSTAG Uplink Tracking



Core Connectivity Failure

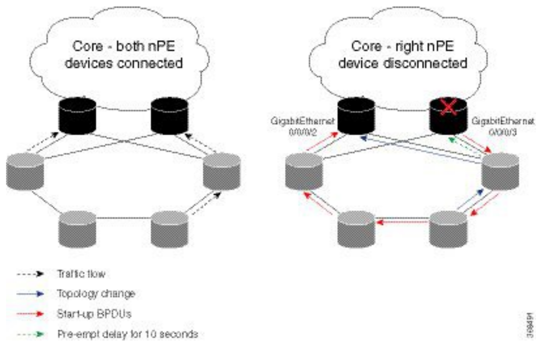
The following diagram illustrates how the nPE device sends start-up BPDUs when it loses core connectivity.

Figure 67: Core connectivity Failure


Core - both nPE devices connected
Core - right nPE device disconnected
Good
Good
Good
Bad
Start-up BPDUs

The device that loses core connectivity starts sending start-up BPDUs. The following diagram illustrates how traffic flows in the access network when an nPE device loses core connectivity.

Figure 68: Active Topology Change


Core - both nPE devices connected
Core - right nPE device disconnected
GigabitEthernet 0/0/0/2
GigabitEthernet 0/0/0/3
Traffic flow
Topology change
Start-up BPDUs
Pre-empt delay for 10 seconds

Start-up BPDUs are sent from the disconnected router to ensure that it is not used as a path to the core from the access ring. This changes the active topology in the access rings depending on the STP configuration.

Benefits

The MSTAG Uplink Tracking feature has these benefits:

- Reduction in the frequency of traffic outages
- Reduced need for redundancy in connectivity from the gateway to the core network

Prerequisites

- Configure MSTAG

Restrictions

Only physical and bundle interfaces, and their sub interfaces can be tracked for core connectivity. Psuedowires themselves cannot be tracked, only the underlying interface carrying the psuedowire traffic can be tracked.

Configure MSTAG Uplink Tracking

Only physical and bundle interfaces, and their sub-interfaces can be tracked for core connectivity. Pseudowires themselves cannot be tracked, only the underlying interface carrying the pseudowire traffic can be tracked.

Perform this task to configure MSTAG Uplink Tracking feature.

```
/* Configure MSTAG for a protocol instance called 'foo', with two interfaces facing the
access ring and pre-empt delay of ten seconds */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# spanning-tree mstag foo
RP/0/RSP0/CPU0:router(config-mstag)# preempt delay for 10 seconds
RP/0/RSP0/CPU0:router(config-mstag)# interface GigabitEthernet0/0/0/0
RP/0/RSP0/CPU0:router(config-mstag-if)# exit
RP/0/RSP0/CPU0:router(config-mstag)# interface GigabitEthernet0/0/0/1
RP/0/RSP0/CPU0:router(config-mstag-if)# commit
RP/0/RSP0/CPU0:router(config-mstag-if)# root

/* Configure Uplink Tracking by adding core-facing interfaces under the 'track' keyword */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# spanning-tree mstag foo
RP/0/RSP0/CPU0:router(config-mstag)#track
RP/0/RSP0/CPU0:router(config-mstag-track)# interface GigabitEthernet0/0/0/2
RP/0/RSP0/CPU0:router(config-mstag-track)# interface GigabitEthernet0/0/0/3
RP/0/RSP0/CPU0:router(config-mstag-track)# commit

/* When pre-empt timer is zero, traffic flows through interface GigabitEthernet0/0/0/3 as
soon as this interface comes up.*/

/* In this configuration, pre-empt delay is set to 10 seconds. Traffic flows through interface
 GigabitEthernet0/0/0/3, ten seconds after this interface comes up. */
```

Running Configuration

```
configure
  spanning-tree mstag foo
    interface GigabitEthernet 0/0/0/0
    !
    interface GigabitEthernet 0/0/0/1
    !
    track
      interface GigabitEthernet 0/0/0/2
      interface GigabitEthernet 0/0/0/3
    !
    pre-empt delay for 10 seconds
  !
!
```

Verification

Verify the core connectivity status. The following example shows the state of the interfaces that connect the core.

```
RP/0/0/CPU0:ios#show spanning-tree mstag foo tracked
Core Connectivity Available: True
Tracked Items:               1/2 up

Interface Name                       State
-----------------------------------------------
GigabitEthernet0/0/0/3               Down
GigabitEthernet0/0/0/2               Up
-----------------------------------------------
```

Configuring PVSTAG or PVRSTAG

This section describes the procedures for configuring PVSTAG:

The procedures for configuring PVRSTAG are identical.

Note This section does not describe how to configure data switching. Refer to the *Implementing Multipoint Layer 2 Services* module for more information.

Enabling PVSTAG

PVSTAG is enabled for a particular VLAN, on a physical interface, by explicit configuration of that physical interface and VLAN for PVSTAG.

Configuring PVSTAG parameters

The configurable PVSTAG parameters for each interface on each VLAN are:

- Root Priority and ID
- Root cost
- Bridge Priority and ID
- Port priority and ID
- Max Age
- Hello Time

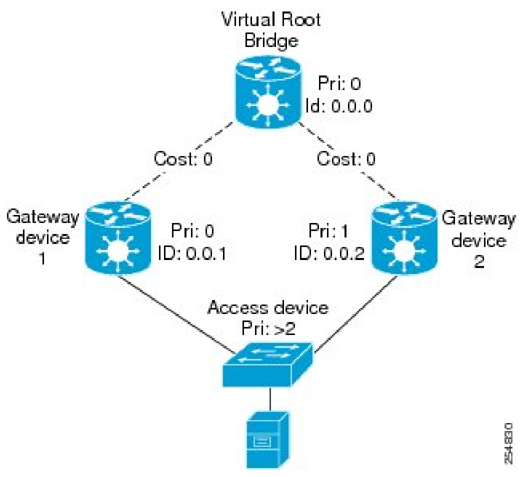
For correct operation, these guidelines must be followed when configuring PVSTAG.

- Both gateway devices should be configured with a root bridge priority and ID that is better (lower) than the bridge priority and Bridge ID of any device in the access network. It is recommended that you set the root bridge priority and ID to 0 on the gateway devices.
- Both gateway devices should be configured with a root cost of 0.
- One gateway device should be configured with the bridge priority and ID that is higher than the root bridge priority and ID, but lower than the bridge priority and ID of any other device in the network (including the other gateway device). It is recommended that you set the bridge priority to 0.
- The second gateway device should be configured with a bridge priority and ID that is higher than the root bridge priority and ID and the first gateway device bridge priority and ID, but lower than the bridge priority and ID of any device in the access network. It is recommended that you set the bridge priority to 1 for PVSTAG or 4096 for PVRSTAG. (For PVRSTAG, this is the lowest allowable value greater than 0.)
- All access devices must be configured with a higher bridge priority than the gateway devices. It is recommended that you use values of 2 or higher for PVSTAG, or 8192 or higher for PVRSTAG.
- For each spanning tree instance, the port path cost and other parameters may be configured on the access devices, so as to ensure the desired port is placed into the blocked state when all links are up.

Caution There are no checks on PVSTAG configuration—misconfiguration may result in incorrect operation of the PVST protocol in the access devices (for example, an STP dispute being detected).

These guidelines are illustrated in the following figure:

Figure 69: PVSTAG Guidelines



Note The configuration steps listed in the following sections show all of the configurable parameters. However, in general, most of these can be retained with the default values.

PVSTAG Topology Restrictions

These restrictions are applicable to PVSTAG topology:

- Only a single access device can be attached to the gateway devices.
- Topology change notifications on a single VLAN affect all VLANs and bridge domains on that physical interface.

SUMMARY STEPS

1. **configure**
2. **spanning-tree mstag pvstag** *protocol instance identifier*
3. **preempt delay for** *interval* { **seconds | minutes | hours** }
4. **interface type** *interface-path-id* or **interface Bundle-Ether** *bundle-id*
5. **vlan** *vlan-id*
6. **root-priority** *priority*
7. **root-id** *id*
8. **root-cost** *cost*
9. **priority** *priority*
10. **bridge-id** *id*
11. **port-priority** *priority*

12. **port-id** *id*
13. **hello-time** *seconds*
14. **max age** *seconds*
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **spanning-tree mstag pvstag** *protocol instance identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# spanning-tree pvstag a
RP/0/RSP0/CPU0:router(config-pvstag)#
```

Enters the PVSTAG configuration submode.

Step 3 **preempt delay for** *interval* { **seconds | minutes | hours** }

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag)# preempt delay for 10 seconds
```

Specifies the delay period during which startup BPDUs should be sent, before preempting.

Step 4 **interface type** *interface-path-id* or **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag)# interface GigabitEthernet0/2/0/30.1
RP/0/RSP0/CPU0:router(config-pvstag-if)#

or

RP/0/RSP0/CPU0:router(config-pvstag)# interface Bundle-Ether 100
RP/0/RSP0/CPU0:router(config-pvstag-if)#
```

Enters the PVSTAG interface configuration submode, and enables PVSTAG for the specified port.

Step 5 **vlan** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if)# vlan 200
```

Enables and configures a VLAN on this interface.

Step 6 **root-priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# root-priority 4096
```

Sets the root bridge priority for the BPDUs sent from this port.

Step 7 **root-id** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# root-id 0000.0000.0000
```

Sets the identifier of the root bridge for BPDUs sent from a port.

Step 8 **root-cost** *cost*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# root-cost 10000
```

Set the root path cost to sent in BPDUs from this interface.

Step 9 **priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# priority 4096
```

Sets the bridge priority for the current MSTI.

For PVSTAG, allowed values are from are 0 through 65535; for PVRSTAG, the allowed values are from 0 through 61440 in multiples of 4096.

Step 10 **bridge-id** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# bridge-id 001c.0000.0011
```

Sets the bridge ID for the current switch.

Step 11 **port-priority** *priority*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# port-priority 160
```

Sets the port priority performance parameter for the MSTI.

For PVSTAG, allowed values for port priority are from 0 through 255; for PVRSTAG, the allowed values are from 0 through 240 in multiples of 16.

Step 12 **port-id** *id*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# port-id 111
```

Sets the port ID for the current switch.

Step 13 **hello-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# hello-time 1
```

Sets the port hello time in seconds.

Allowed values are from 1 through 2.

Step 14 **max age** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-pvstag-if-vlan)# max age 20
```

Sets the maximum age performance parameters for the bridge.

Allowed values for the maximum age time for the bridge in seconds are from 6 through 40.

Note Repeat steps 4 to 14 to configure each interface; repeat steps 5 to 14 to configure each VLAN on each interface.

Step 15 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Subinterfaces

For each VLAN that is enabled for PVSTAG on an interface, a corresponding subinterface that matches traffic for that VLAN must be configured. This is used both for data switching and for PVST BPDUs. Follow these guidelines when configuring subinterfaces:

- VLAN 1 is treated as the native VLAN in PVST. Therefore, for VLAN 1, a subinterface that matches untagged packets (**encapsulation untagged**) must be configured. It may also be necessary to configure a subinterface that matches packets tagged explicitly with VLAN 1 (**encapsulation dot1q 1**).
- Only dot1q packets are allowed in PVST; Q-in-Q and dot1ad packets are not supported by the protocol, and therefore subinterfaces configured with these encapsulation will not work correctly with PVSTAG.
- Subinterfaces that match a range of VLANs are supported by PVSTAG; it is not necessary to configure a separate subinterface for each VLAN, unless it is desirable for provisioning the data switching.

- PVSTAG does not support:
    - Physical interfaces configured in L2 mode
    - Subinterface configured with a default encapsulation (**encapsulation default**)
    - Subinterfaces configured to match any VLAN (**encapsulation dot1q any**)

For more information about configuring L2 subinterfaces, refer to the Implementing Point to Point Layer 2 Services module.

Verifying PVSTAG

These show commands allow you to verify the operation of PVSTAG or PVRSTAG:

- **show spanning-tree pvstag** *mst-name*
- **show spanning-tree pvstag** *mst-name*

In particular, these commands display the subinterface that is being used for each VLAN.

Configuring PVRST

Before you begin

Ensure that:

- L2transport subinterfaces with VLAN encapsulation are defined.
- L2VPN bridge domains under bridge group for every VLAN running spanning tree are configured, and corresponding l2transport subinterfaces are configured in the bridge-domain.

SUMMARY STEPS

1. **configure**
2. **spanning-tree pvrst** *protocol-instance-name*
3. **apply-group** *group_name group_name*
4. **forward-delay** *seconds*
5. **maximum age** *seconds*
6. **transmit hold-count** *count*
7. **vlan** *vlan_id*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **spanning-tree pvrst** *protocol-instance-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# spanning-tree pvrst stp
```

Enters the PVRST configuration submode.

Step 3 **apply-group** *group_name group_name*

Example:

```
RP/0/RSP0/CPU0:router(config-pvrst)# apply-group groupA groupB
```

Allows you to apply configuration from one group to another.

Step 4 **forward-delay** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-pvrst)# forward-delay 10
```

Allows you to configure bridge forward delay time in seconds.

The forward delay is the number of seconds a port waits before changing from its spanning-tree learning and listening states to the forwarding state. The delay time range is from 4 to 30.

Step 5 **maximum age** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-pvrst)# maximum age 10
```

Specifies maximum age for the bridge in seconds.

The maximum-aging time is the number of seconds a switch waits without receiving spanning-tree configuration messages before attempting a reconfiguration. The maximum age range is from 6 to 40 seconds.

Step 6 **transmit hold-count** *count*

Example:

```
RP/0/RSP0/CPU0:router(config-pvrst)# transmit hold-count 4
```

Allows you to configure bridge transmit hold count. The hold count range is from 1 to 10.

Step 7 **vlan** *vlan_id*

Example:

```
RP/0/RSP0/CPU0:router(config-pvrst)#
```

Allows you to configure PVRST on a VLAN. The VLAN ID range is from 1 to 4094.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MVRP-lite

This section describes the procedure for configuring MVRP-lite:

Enabling MVRP-lite

When MVRP-lite is configured, it is automatically enabled on all interfaces where MSTP is enabled. MSTP must be configured before MVRP can be enabled. For more information on configuring MSTP, see Configuring MSTP.

Configuring MVRP-lite parameters

The configurable MVRP-lite parameters are:

- Periodic Transmission
- Join Time
- Leave Time
- Leave-all Time

SUMMARY STEPS

1. **configure**
2. **spanning-tree mst** *protocol instance identifier*
3. **mvrp static**
4. **periodic transmit** [**interval** *seconds*]
5. **join-time** *milliseconds*
6. **leave-time** *seconds*
7. **leaveall-time** *seconds*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **spanning-tree mst** *protocol instance identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)#
spanning-tree mst aRP/0/RSP0/CPU0:router(config-mstp)#
```

Enters the MSTP configuration submode.

Step 3 **mvrp static**

Example:

```
RP/0/RSP0/CPU0:router(config-mstp)#mvrp static
```

Configures MVRP to run over this MSTP protocol instance.

Step 4 **periodic transmit** [**interval** *seconds*]

Example:

```
RP/0/RSP0/CPU0:router(config-mvrp)#
periodic transmit
```

Sends periodic Multiple VLAN Registration Protocol Data Unit (MVRPDU) on all active ports.

Step 5 **join-time** *milliseconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mvrp)#
 hello-time 1
```

Sets the join time for all active ports.

Step 6 **leave-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mvrp)# leave-time 20
```

Sets the leave time for all active ports.

Step 7 **leaveall-time** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-mvrp)# leaveall-time 20
```

Sets the leave all time for all active ports.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Verifying MVRP-lite

These show commands allow you to verify the operation of MVRP-lite:

- **show ethernet mvrp mad**
- **show ethernet mvrp status**
- **show ethernet mvrp statistics**

Configuration Examples for Implementing MSTP

This section provides configuration examples for the following:

Configuring MSTP: Examples

This example shows MSTP configuration for a single spanning-tree instance with MSTP enabled on a single interface:

```
config
spanning-tree mst example
    name m1
    revision 10
    forward-delay 20
    maximum hops 40
    maximum age 40
    transmit hold-count 8
    provider-bridge
    bringup delay for 60 seconds
    flush containment disable
    instance 101
        vlans-id 101-110
        priority 8192
    !
    interface GigabitEthernet0/0/0/0
        hello-time 1
        external-cost 10000
        link-type point-to-point
        portfast
        guard root
        guard topology-change
        instance 101 cost 10000
        instance 101 port-priority 160
    !
!
```

This example shows the output from the **show spanning-tree mst** command, which produces an overview of the spanning tree protocol state:

```
# show spanning-tree mst example
Role:  ROOT=Root, DSGN=Designated, ALT=Alternate, BKP=Backup, MSTR=Master
State: FWD=Forwarding, LRN=Learning, BLK=Blocked, DLY=Bringup Delayed

Operating in dot1q mode


MSTI 0 (CIST):

  VLANS Mapped: 1-9,11-4094

  CIST Root  Priority    4096
             Address     6262.6262.6262
             This bridge is the CIST root
             Ext Cost    0

  Root ID    Priority    4096
             Address     6262.6262.6262
             This bridge is the root
             Int Cost    0
             Max Age 20 sec, Forward Delay 15 sec


  Bridge ID  Priority    4096 (priority 4096 sys-id-ext 0)
             Address     6262.6262.6262
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6
Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
Gi0/0/0/0    128.1   20000     DSGN FWD   4096  6262.6262.6262 128.1
Gi0/0/0/1    128.2   20000     DSGN FWD   4096  6262.6262.6262 128.2
Gi0/0/0/2    128.3   20000     DSGN FWD   4096  6262.6262.6262 128.3
Gi0/0/0/3    128.4   20000     ---- BLK   ----- -------------- -------


MSTI 1:

  VLANS Mapped: 10

  Root ID    Priority    4096
             Address     6161.6161.6161
             Int Cost    20000
             Max Age 20 sec, Forward Delay 15 sec


  Bridge ID  Priority    32768 (priority 32768 sys-id-ext 0)
             Address     6262.6262.6262
             Max Age 20 sec, Forward Delay 15 sec
             Max Hops 20, Transmit Hold count  6


Interface    Port ID           Role State Designated           Port ID
             Pri.Nbr Cost                 Bridge ID            Pri.Nbr
------------ ------- --------- ---- ----- -------------------- -------
Gi0/0/0/0    128.1   20000     ROOT FWD   4096  6161.6161.6161 128.1
Gi0/0/0/1    128.2   20000     ALT  BLK   4096  6161.6161.6161 128.2
Gi0/0/0/2    128.3   20000     DSGN FWD   32768 6262.6262.6262 128.3
Gi0/0/0/3    128.4   20000     ---- BLK   ----- -------------- -------

==========================================================================
```

In the **show spanning-tree mst** example output, the first line indicates whether MSTP is operating in dot1q or the Provider Bridge mode, and this information is followed by details for each MSTI.

For each MSTI, the following information is displayed:

- The list of VLANs for the MSTI.
- For the CIST, the priority and bridge ID of the CIST root, and the external path cost to reach the CIST root. The output also indicates if this bridge is the CIST root.
- The priority and bridge ID of the root bridge for this MSTI, and the internal path cost to reach the root. The output also indicates if this bridge is the root for the MSTI.
- The max age and forward delay times received from the root bridge for the MSTI.
- The priority and bridge ID of this bridge, for this MSTI.
- The maximum age, forward delay, max hops and transmit hold-count for this bridge (which is the same for every MSTI).
- A list of MSTP-enabled interfaces. For each interface, the following information is displayed:
    - The interface name
    - The port priority and port ID for this interface for this MSTI.
    - The port cost for this interface for this MSTI.
    - The current port role:
        - DSGN—Designated: This is the designated port on this LAN, for this MSTI
        - ROOT—Root: This is the root port for the bridge for this MSTI.
        - ALT—Alternate: This is an alternate port for this MSTI.
        - BKP—Backup: This is a backup port for this MSTI
        - MSTR—Master: This is a boundary port that is a root or alternate port for the CIST.

      The interface is down, or the bringup delay timer is running and no role has been assigned yet.
    - The current port state:
        - BLK—The port is blocked.
        - LRN—The port is learning.
        - FWD—The port is forwarding.
        - DLY—The bringup-delay timer is running.
    - If the port is a boundary port, and not CIST and the port is not designated, then only the BOUNDARY PORT is displayed and the remaining information is not displayed.
    - If the port is not up, or the bringup delay timer is running, no information is displayed for the remaining fields. Otherwise, the bridge priority and bridge ID of the designated bridge on the LAN that the interface connects to is displayed, followed by the port priority and port ID of the designated port on the LAN. If the port role is Designated, then the information for this bridge or port is displayed.

The following example shows the output from the **show spanning-tree mst** command, which produces more detailed information regarding interface state than the standard command as described above:

```
# show spanning-tree mst a interface GigabitEthernet0/1/2/1
GigabitEthernet0/1/2/1
Cost: 20000
link-type: point-to-point
hello-time 1
Portfast: no
BPDU Guard: no
Guard root: no
Guard topology change: no
BPDUs sent 492, received 3

MST 3:
 Edge port:
 Boundary : internal
 Designated forwarding
 Vlans mapped to MST 3: 1-2,4-2999,4000-4094
 Port info port id 128.193 cost 200000
 Designated root address 0050.3e66.d000 priority 8193 cost 20004
 Designated bridge address 0002.172c.f400 priority 49152 port id 128.193
 Timers: message expires in 0 sec, forward delay 0, forward transitions 1
 Transitions to reach this state: 12
```

The output includes interface information about the interface which applies to all MSTIs:

- Cost
- link-type
- hello-time
- portfast (including whether BPDU guard is enabled)
- guard root
- guard topology change
- BPDUs sent, received.

It also includes information specific to each MSTI:

- Port ID, priority, cost
- BPDU information from root (bridge ID, cost, and priority)
- BPDU information being sent on this port (Bridge ID, cost, priority)
- State transitions to reach this state.
- Topology changes to reach this state.
- Flush containment status for this MSTI.

This example shows the output of **show spanning-tree mst** errors, which produces information about interfaces that are configured for MSTP but where MSTP is not operational. Primarily this shows information about interfaces which do not exist:

```
# show spanning-tree mst a errors
Interface               Error
------------------------------
GigabitEthernet1/2/3/4   Interface does not exist.
```

This example shows the output of **show spanning-tree mst** configuration, which displays the VLAN ID to MSTI mapping table. It also displays the configuration digest which is included in the transmitted BPDUs—this must match the digest received from other bridges in the same MSTP region:

```
# show spanning-tree mst a configuration
Name          leo
Revision      2702
Config Digest 9D-14-5C-26-7D-BE-9F-B5-D8-93-44-1B-E3-BA-08-CE
Instance  Vlans mapped
--------  ------------------------------
0         1-9,11-19,21-29,31-39,41-4094
1         10,20,30,40
-----------------------------------------
```

This example shows the output of **show spanning-tree mst**, which produces details on the BPDUs being output and received on a given local interface:

Note Several received packets can be stored in case of MSTP operating on a shared LAN.

```
# show spanning-tree mst a bpdu interface GigabitEthernet0/1/2/2 direction
 transmit
MSTI 0 (CIST):
 Root ID : 0004.9b78.0800
 Path Cost : 83
 Bridge ID : 0004.9b78.0800
 Port ID : 12
 Hello Time : 2
 ...
```

This example shows the output of **show spanning-tree mst**, which displays details about the topology changes that have occurred for each MSTI on each interface:

```
# show spanning-tree mst M topology-change flushes instance$
MSTI 1:

Interface     Last TC               Reason                           Count
------------  --------------------  -------------------------------- -----
Te0/0/0/1     04:16:05 Mar 16 2010  Role change: DSGN to ----           10
#
#
# show spanning-tree mst M topology-change flushes instance$
MSTI 0 (CIST):

Interface     Last TC               Reason                           Count
------------  --------------------  -------------------------------- -----
Te0/0/0/1     04:16:05 Mar 16 2010  Role change: DSGN to ----           10
#
```

Configuring MSTAG: Examples

This example shows MSTAG configuration for a single spanning-tree instance on a single interface:

```
config
interface GigabitEthernet0/0/0/0.1 l2transport
    encapsulation untagged
!
spanning-tree mstag example
    preempt delay for 60 seconds
    interface GigabitEthernet0/0/0/0.1
        name m1
        revision 10
        external-cost 0
        bridge-id 0.0.1
        port-id 1
        maximum age 40
        provider-bridge
        hello-time 1
        instance 101
              edge-mode
            vlans-id 101-110
            root-priority 0
            root-id 0.0.0
            cost 0
            priority 0
            port-priority 0
        !
    !
!
```

This example shows additional configuration for MSTAG Topology Change Propagation:

```
l2vpn
    xconnect group example
        p2p mstag-example
            interface GigabitEthernet0/0/0/0.1
            neighbor 123.123.123.1 pw-id 100
        !
    !
!
```

This example shows the output of **show spanning-tree mstag**:

```
# show spanning-tree mstag A
GigabitEthernet0/0/0/1
  Preempt delay is disabled.
  Name:            6161:6161:6161
  Revision:        0
  Max Age:         20
  Provider Bridge: no
  Bridge ID:       6161.6161.6161
  Port ID:         1
  External Cost:   0
  Hello Time:      2
  Active:          no
  BPDUs sent:      0
    MSTI 0 (CIST):
    VLAN IDs:         1-9,32-39,41-4094
    Role:             Designated
    Bridge Priority:  32768
    Port Priority:    128
    Cost:             0
    Root Bridge:      6161.6161.6161
    Root Priority:    32768
    Topology Changes: 123
  MSTI 2
    VLAN IDs:         10-31
    Role:             Designated
    Bridge Priority:  32768
    Port Priority:    128
    Cost:             0
    Root Bridge:      6161.6161.6161
    Root Priority:    32768
    Topology Changes: 123
  MSTI 10
 VLAN IDs:         40
    Role:             Root (Edge mode)
    Bridge Priority:  32768
    Port Priority:    128
    Cost:             200000000
    Root Bridge:      6161.6161.6161
    Root Priority:    61440
    Topology Changes: 0
```

This example shows the output of **show spanning-tree mstag bpdu interface**, which produces details on the BPDUs being output and received on a given local interface:

```
RP/0/RSP0/CPU0:router#show spanning-tree mstag foo bpdu interface GigabitEthernet 0/0/0/0
Transmitted:
  MSTI 0 (CIST):
ProtocolIdentifier: 0
ProtocolVersionIdentifier: 3
BPDUType: 2
CISTFlags: Top Change Ack  0
           Agreement       1
           Forwarding      1
           Learning        1
           Role            3
           Proposal        0
           Topology Change 0
CISTRootIdentifier: priority 8, MSTI 0, address 6969.6969.6969
CISTExternalPathCost: 0
CISTRegionalRootIdentifier: priority 8, MSTI 0, address 6969.6969.6969
CISTPortIdentifierPriority: 8
CISTPortIdentifierId: 1
MessageAge: 0
MaxAge: 20
HelloTime: 2
ForwardDelay: 15
Version1Length: 0
Version3Length: 80
FormatSelector: 0
Name: 6969:6969:6969
Revision: 0
MD5Digest: ac36177f 50283cd4 b83821d8 ab26de62
CISTInternalRootPathCost: 0
CISTBridgeIdentifier: priority 8, MSTI 0, address 6969.6969.6969
CISTRemainingHops: 20
  MSTI 1:
MSTIFlags: Master          0
           Agreement       1
           Forwarding      1
           Learning        1
           Role            3
           Proposal        0
           Topology Change 0
MSTIRegionalRootIdentifier: priority 8, MSTI 1, address 6969.6969.6969
MSTIInternalRootPathCost: 0
MSTIBridgePriority: 1
MSTIPortPriority: 8
MSTIRemainingHops: 20
```

This example shows the output of **show spanning-tree mstag topology-change flushes**, which displays details about the topology changes that have occurred for each interface:

```
#show spanning-tree mstag b topology-change flushes

MSTAG Protocol Instance b
```

```
Interface     Last TC             Reason                           Count
------------  -------------------  -------------------------------- -----
Gi0/0/0/1     18:03:24 2009-07-14  Gi0/0/0/1.10 egress TCN          65535
Gi0/0/0/2     21:05:04 2009-07-15  Gi0/0/0/2.1234567890 ingress TCN     2
```

Configuring PVSTAG: Examples

This example shows PVSTAG configuration for a single VLAN on a single interface:

```
config
spanning-tree pvstag example
    preempt delay for 60 seconds
    interface GigabitEthernet0/0/0/0
        vlan 10
            root-priority 0
            root-id 0.0.0
            root-cost 0
            priority 0
            bridge-id 0.0.1
            port-priority 0
            port-id 1
            max age 40
            hello-time 1
        !
    !
!
```

This example shows the output of **show spanning-tree pvstag**:

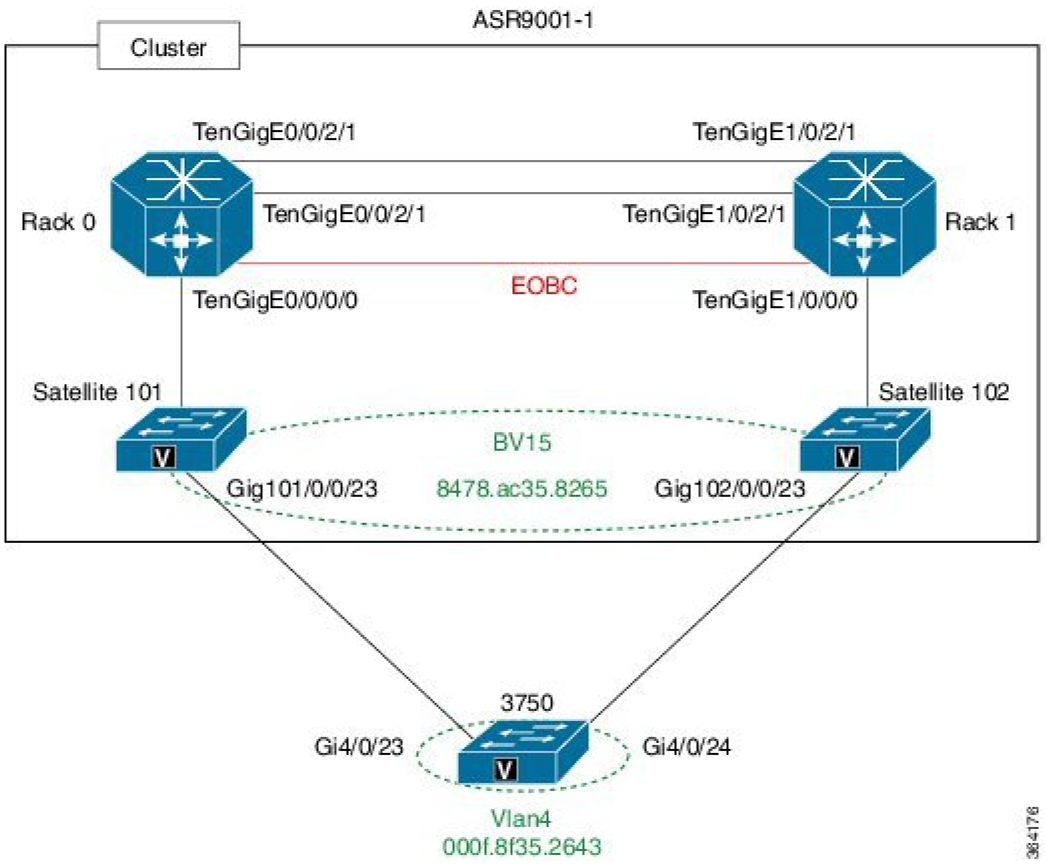
```
# show spanning-tree pvstag interface GigabitEthernet0/0/0/1
GigabitEthernet0/0/0/1
  VLAN 10
    Preempt delay is disabled.
    Sub-interface:    GigabitEthernet0/0/0/1.20 (Up)
    Max Age:          20
    Root Priority:    0
    Root Bridge:      0000.0000.0000
    Cost:             0
    Bridge Priority:  32768
    Bridge ID:        6161.6161.6161
    Port Priority:    128
    Port ID:          1
    Hello Time:       2
    Active:           no
    BPDUs sent:       0
    Topology Changes: 123
    VLAN 20
```

Configuring PVSTAG on a Cluster with a Satellite: Example

This example shows application of PVSTAG in a cluster illustrated in the following network topology diagram, For higher scale fanout, the cluster has satellite switches. The cluster provides L3 termination through a routed interface (BVI). There is no VPLS in the core. PVSTAG is configured in this scenario with the single purpose of eliminating full spanning tree from the cluster.

To simplify this PVSTAG example, there is only one access switch in the network topology. In a real scenario, you can replace this with a ring of access switches.

Figure 70: PVSTAG on a Cluster with a Satellite



Cluster Configuration:

```
!
spanning-tree pvstag inst4
interface GigabitEthernet101/0/0/23
vlan 4
root-priority 4096
root-id 0011.0011.0011
priority 8192
bridge-id 0011.0011.0011
port-priority 128
!
!
interface GigabitEthernet102/0/0/23
vlan 4
root-priority 4096
root-id 0011.0011.0011
priority 8192
bridge-id 0022.0022.0022
port-priority 128
!
!
!
interface Loopback100
ipv4 address 8.8.8.8 255.255.255.255
!
interface Loopback201
ipv4 address 9.9.9.9 255.255.255.255
```

```
!
interface GigabitEthernet101/0/0/23
cdp
!
interface GigabitEthernet102/0/0/23
cdp
!
interface GigabitEthernet101/0/0/23.4 l2transport
encapsulation dot1q 4
rewrite ingress tag pop 1 symmetric
!
interface GigabitEthernet102/0/0/23.4 l2transport
encapsulation dot1q 4
rewrite ingress tag pop 1 symmetric
!
interface TenGigE0/0/0/0
ipv4 point-to-point
ipv4 unnumbered Loopback100
nv
satellite-fabric-link satellite 101
remote-ports GigabitEthernet 0/0/0-43
!
!
!
interface TenGigE0/0/2/0
nv
edge
interface
!
!
!
interface TenGigE0/0/2/1
nv
edge
interface
!
!
!
interface TenGigE1/0/0/0
ipv4 point-to-point
ipv4 unnumbered Loopback201
nv
satellite-fabric-link satellite 102
remote-ports GigabitEthernet 0/0/0-43
!
!
!
interface BVI5
ipv4 address 4.4.4.1 255.255.255.0
!
l2vpn
bridge group bg1
bridge-domain bd1
interface GigabitEthernet101/0/0/23.4
!
interface GigabitEthernet102/0/0/23.4
!
routed interface BVI5
!
!
!
nv
satellite 101
type asr9000v
```

```
serial-number CAT1641U0QV
ipv4 address 10.22.1.2
!
satellite 102
type asr9000v
serial-number CAT1635U14B
ipv4 address 10.23.1.2
!
```

3750 Configuration:

```
!
interface GigabitEthernet4/0/23
switchport trunk encapsulation dot1q
switchport trunk allowed vlan 4
switchport mode trunk
!
interface GigabitEthernet4/0/24
switchport trunk encapsulation dot1q
switchport trunk allowed vlan 4
switchport mode trunk
!
interface Vlan4
ip address 4.4.4.2 255.255.255.0
!
```

Configuring PVRST: Example

This example shows a sample PVRST configuration.

```
(config)# spanning-tree pvrst stp1
(config-pvrst)# forward-delay 6
(config-pvrst)# interface GigabitEthernet 0/1/1/2 hello-time 2
(config-pvrst)# maximum age 35
(config-pvrst)# transmit hold-count 9
(config-pvrst)# vlan 666 priority 4096
(config-pvrst)# commit
```

Configuring MVRP-Lite: Examples

This example shows MVRP-lite configuration:

```
config
spanning-tree mst example
    mvrp static
        periodic transmit
        join-time 200
        leave-time 30
        leaveall-time 10
    !
!
```

This example shows the output of **show ethernet mvrp mad**:

```
RP/0/RSP0/CPU0:router# show ethernet mvrp mad interface GigabitEthernet 0/1/0/1
GigabitEthernet0/1/0/1
  Participant Type: Full; Point-to-Point: Yes
  Admin Control: Applicant Normal; Registrar Normal

  LeaveAll Passive (next in 5.92s); periodic disabled
  Leave in 25.70s; Join not running
  Last peer 0293.6926.9585; failed registrations: 0

VID   Applicant              Registrar
----  ---------------------  ---------
   1  Very Anxious Observer  Leaving
 283  Quiet Passive          Empty
```

This example shows the output of **show ethernet mvrp status**:

```
RP/0/RSP0/CPU0:router# show ethernet mvrp status interface GigabitEthernet 0/1/0/1
GigabitEthernet0/1/0/1
  Statically declared:  1-512,768,980-1034
  Dynamically declared: 2048-3084
  Registered:           1-512
```

This example shows the output of **show ethernet mvrp statistics**:

```
RP/0/RSP0/CPU0:router# show ethernet mvrp statistics interface GigabitEthernet 0/1/0/1
GigabitEthernet0/1/0/1
  MVRPDUs TX:    1245
  MVRPDUs RX:       7
  Dropped TX:       0
  Dropped RX:      42
  Invalid RX:      12
```

CHAPTER 9

Implementing of Layer 2 Access Lists

An Ethernet services access control list (ACL) consists of one or more access control entries (ACE) that collectively define the Layer 2 network traffic profile. This profile can then be referenced by Cisco IOS XR software features. Each Ethernet services ACL includes an action element (permit or deny) based on criteria such as source and destination address, Class of Service (CoS), or VLAN ID.

This module describes tasks required to implement Ethernet services access lists on your Cisco ASR 9000 Series Aggregation Services Router.

Note For a complete description of the Ethernet services access list commands listed in this module, refer to the *Ethernet Services (Layer 2) Access List Commands on Cisco ASR 9000 Series Routers* module in the *Cisco ASR 9000 Series Aggregation Services Router IP Addresses and Services Command Reference* publication.

Feature History for Implementing Ethernet Services Access Lists on Cisco ASR 9000 Series Routers

Release	Modification
Release 3.7.2	This feature was introduced on Cisco ASR 9000 Series Routers.

Prerequisites for Implementing Layer 2 Access Lists

This prerequisite applies to implement access lists and prefix lists:

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing Layer 2 Access Lists

Ethernet Services Access Lists Feature Highlights

Ethernet services access lists have these feature highlights:

- The ability to clear counters for an access list using a specific sequence number.
- The ability to copy the contents of an existing access list to another access list.
- Allows users to apply sequence numbers to permit or deny statements and to resequence, add, or remove such statements from a named access list.
- Provides packet filtering on interfaces to forward packets.
- Ethernet services ACLs can be applied on interfaces, VLAN subinterfaces, bundle-Ethernet interfaces, EFPs, and EFPs over bundle-Ethernet interfaces. Atomic replacement of Ethernet services ACLs is supported on these physical interfaces.

Purpose of Ethernet Services Access Lists

Using ACL-based forwarding (ABF), Ethernet services access lists perform packet filtering to control which packets move through the network and where. Such controls help to limit incoming and outgoing network traffic and restrict the access of users and devices to the network at the port level.

How an Ethernet Services Access List Works

An Ethernet services access list is a sequential list consisting of permit and deny statements that apply to Layer 2 configurations. The access list has a name by which it is referenced.

An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control Layer 2 traffic arriving at the router or leaving the router, but not traffic originating at the router.

Ethernet Services Access List Process and Rules

Use this process and rules when configuring an Ethernet services access list:

- The software tests the source or destination address of each packet being filtered against the conditions in the access list, one condition (permit or deny statement) at a time.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the remaining statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies the address or protocol, the software discards the packet.

- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.
- The access list should contain at least one permit statement or else all packets are denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- Inbound access lists process packets arriving at the router. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; deny means discard the packet.
- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; deny means discard the packet.
- An access list can not be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.
- An access list must exist before you can use the **ethernet-services access-group** command.

Helpful Hints for Creating Ethernet Services Access Lists

Consider these when creating an Ethernet services access list:

- Create the access list before applying it to an interface.
- Organize your access list so that more specific references appear before more general ones.

Source and Destination Addresses

Source MAC address and destination MAC address are two of the most typical fields on which to base an access list. Specify source MAC addresses to control packets from certain networking devices or hosts. Specify destination MAC addresses to control packets being sent to certain networking devices or hosts.

Ethernet Services Access List Entry Sequence Numbering

The ability to apply sequence numbers to Ethernet services access-list entries simplifies access list changes. The access list entry sequence numbering feature allows you to add sequence numbers to access-list entries and resequence them. When you add a new entry, you choose the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Sequence Numbering Behavior

These details the sequence numbering behavior:

- If entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483646. If the generated sequence number exceeds this maximum number, this message is displayed:

```
Exceeded maximum sequence number.
```

- If you provide an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- ACL entries can be added without affecting traffic flow and hardware performance.
- Distributed support is provided so that the sequence numbers of entries in the route-switch processor (RSP) and interface card are synchronized at all times.

How to Implement Layer 2 Access Lists

Restrictions for Implementing Layer 2 Access Lists

These restrictions apply for implementing Ethernet services access lists:

- Ethernet services access lists are not supported over management interfaces.
- NetIO (software slow path) is not supported for Ethernet services access lists.
- Match on inner VLAN 0 and outer VLAN 0 is not supported on Cisco ASR 9000 High Density 100GE Ethernet Line Card and ASR 9000 Enhanced Ethernet Line Card.

Configuring Ethernet Services Access Lists

This task configures an Ethernet services access list.

SUMMARY STEPS

1. **configure**
2. **ethernet-services access-list** *name*
3. [*sequence-number*] **{ permit | deny }** { *src-mac-address src-mac-mask* | **any** | **host** } [{ *ethertype-number* } | **vlan** *min-vlan-ID* [*max-vlan-ID*]] [**cos** *cos-value*] [**dei**] [**inner-vlan** *min-vlan-ID* [*max-vlan-ID*]] **inner-cos** *cos-value*] [**inner-dei**]
4. Repeat Step 3 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
Router# configure
```

Enters the Global Configuration mode.

Step 2 **ethernet-services access-list** *name*

Example:

```
Router(config)# ethernet-services access-list L2ACL2
```

Enters Ethernet services access list configuration mode and configures access list L2ACL2.

Step 3 [*sequence-number*] **{ permit | deny }** { *src-mac-address src-mac-mask* | **any** | **host** } [{ *ethertype-number* } | **vlan** *min-vlan-ID* [*max-vlan-ID*]] [**cos** *cos-value*] [**dei**] [**inner-vlan** *min-vlan-ID* [*max-vlan-ID*]] **inner-cos** *cos-value*] [**inner-dei**]

The following example shows to allow the specified source MAC address and to deny any other source MAC

Example:

```
Router(config-es-al)# 10 permit host 0001.0002.0001 any

or

Router(config-es-al)# 30 deny any any
```

Specifies one or more conditions allowed or denied, which determines whether the packet is passed or dropped.

Step 4 Repeat Step 3 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.

Allows you to revise an access list.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

What to Do Next

After creating an Ethernet services access list, you must apply it to an interface. See the Applying Ethernet Services Access Lists section for information about how to apply an access list.

Applying Ethernet Services Access Lists

After you create an access list, you must reference the access list to make it work. Access lists can be applied on either outbound or inbound interfaces. This section describes guidelines on how to accomplish this task for both terminal lines and network interfaces.

For inbound access lists, after receiving a packet, Cisco IOS XR software checks the source MAC address of the packet against the access list. If the access list permits the address, the software continues to process the packet. If the access list rejects the address, the software discards the packet.

For outbound access lists, after receiving and routing a packet to a controlled interface, the software checks the source MAC address of the packet against the access list. If the access list permits the address, the software sends the packet. If the access list rejects the address, the software discards the packet.

Note An empty access-list (containing no access control elements) cannot be applied on an interface.

Controlling Access to an Interface

This task applies an access list to an interface to restrict access to that interface. Access lists can be applied on either outbound or inbound interfaces.

SUMMARY STEPS

1. **configure**
2. **interface** *type instance*
3. **ethernet-services access-group** *access-list-name* { **ingress** | **egress** }
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
Router# configure
```

Enters the Global Configuration mode.

Step 2 **interface** *type instance*

Example:

```
Router(config)# interface gigabitethernet 0/2/0/2
```

Configures an interface and enters interface configuration mode.

- The *type* argument specifies an interface type. For more information on interface types, use the question mark (?) online help function.
- The *instance* argument specifies either a physical interface instance or a virtual instance.
  - The naming notation for a physical interface instance is *rack/slot/module/port*. The slash (/) between values is required as part of the notation.
  - The number range for a virtual interface instance varies depending on the interface type.

Step 3 **ethernet-services access-group** *access-list-name* { **ingress** | **egress** }

Example:

```
Router(config-if)# ethernet-services access-group p-in-filter ingress

Router(config-if)# ethernet-services access-group p-out-filter egress
```

Controls access to an interface.

- Use the *access-list-name* argument to specify a particular Ethernet services access list.
- Use the *ingress* keyword to filter on inbound packets or the *egress* keyword to filter on outbound packets.

This example applies filters on packets inbound and outbound from GigabitEthernet interface 0/2/0/2.

Step 4 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Copying Ethernet Services Access Lists

This task copies an Ethernet services access list.

SUMMARY STEPS

1. **copy access-list ethernet-service** *source-acl destination-acl*
2. **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

DETAILED STEPS

Step 1 **copy access-list ethernet-service** *source-acl destination-acl*

Example:

```
RP/0/RSP0/CPU0:router# copy access-list ethernet-service list-1 list-2
```

Creates a copy of an existing Ethernet services access list.

- Use the *source-acl* argument to specify the name of the access list to be copied.
- Use the *destination-acl* argument to specify where to copy the contents of the source access list.
  - The *destination-acl* argument must be a unique name; if the destination-acl argument name exists for an access list, the access list is not copied.

Step 2 **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

Example:

```
RP/0/RSP0/CPU0:router# show access-lists ethernet-services list-2
```

(Optional) Displays the contents of a named Ethernet services access list. For example, you can verify the output to see that the destination access list list-2 contains all the information from the source access list list-1.

Resequencing Access-List Entries

This task shows how to reassign sequence numbers to entries in a named access list. Resequencing an access list is optional.

SUMMARY STEPS

1. **resequence access-list ethernet-services** *access-list-name* [*starting-sequence-number* [*increment*]]
2. **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

DETAILED STEPS

Step 1 **resequence access-list ethernet-services** *access-list-name* [*starting-sequence-number* [*increment*]]

Example:

```
Router# resequence access-list ethernet-services L2ACL2 20 10
```

(Optional) Resequences the specified Ethernet services access list using the desired starting sequence number and the increment of sequence numbers.

- This example resequences an Ethernet services access list named L2ACL2. The starting sequence number is 20 and the increment is 10. If you do not select an increment, the default increment 10 is used.

Note If during the resequencing process it is determined that the ending number will exceed the maximum sequence number allowed, the configuration will not take effect and will be rejected. The sequence numbers will not be changed.

Step 2 **show access-lists ethernet-services** [*access-list-name* | **maximum** | **standby** | **summary**]

Example:

```
Router# show access-lists ethernet-services L2ACL2
```

(Optional) Displays the contents of a named Ethernet services access list.

- Review the output to see that the access list includes the updated information.

Configuration Examples for Implementing Layer 2 Access Lists

Resequencing Entries in an Access List: Example

This example shows access-list resequencing. The starting value in the resequenced access list is 1, and the increment value is 2. The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483646.

When an entry with no sequence number is entered, by default, it has a sequence number of 10 more than the last entry in the access list.

```
ethernet-services access-list test_acl3
 10 permit host 0001.0002.0001 any
 13 permit host 0001.0002.0002 any
 16 permit host 0001.0002.0003 any
 19 deny any any
```

```
resequence access-list ethernet-services test_acl3 10 3
```

```
show access-lists ethernet-services test_acl3
```

```
ethernet service access-list acl_1
10 permit 1.2.3 4.5.6
20 deny 2.3.4 5.4.3
30 permit 3.1.2 5.3.4 cos 5

resequence access-list ethernet service acl_1 10 20

show access-list ethernet-service acl1_1

ipv4 access-list acl_1
  10 permit 1.2.3 4.5.6
  30 deny 2.3.4 5.4.3
  50 permit 3.1.2 5.3.4 cos 5
```

Adding Entries with Sequence Numbers: Example

In this example, a new entry is added to Ethernet services access list test_acl3.

```
 10 permit host 0001.0002.0001 any
 13 permit host 0001.0002.0002 any cos 3
 16 permit host 0001.0002.0003 any vlan 10
 19 deny any any
!

configure
ethernet-services access-list test_acl3
15 permit host 0001.0002.0004 any

ethernet-services access-list test_acl3
 10 permit host 0001.0002.0001 any
 13 permit host 0001.0002.0002 any cos 3
 15 permit host 0001.0002.0004 any
 16 permit host 0001.0002.0003 any vlan 10
 19 deny any any
!
```

CHAPTER 10

Implementing VXLAN

This module provides conceptual information for VXLAN in general and configuration information for layer 2 VXLAN on Cisco ASR 9000 Series Router. For configuration information of layer 3 VXLAN, see *Implementing L3 VXLAN* chapter in the *Cisco ASR 9000 Series Aggregation Services Router MPLS Layer 3 VPN Configuration Guide*. VXLAN provides the same Ethernet Layer 2 network services as VLAN, but with greater extensibility and flexibility.

Table 8: Feature History for VXLAN

Release	Modification
Release 5.2.0	This feature was introduced on Cisco ASR 9000 Series Router.
Release 5.3.1	VXLAN Anycast Gateway feature was introduced

Prerequisites for implementing VXLANs

This prerequisite applies to implementing VXLANs:

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information about Implementing VXLAN

To implement VXLAN, you must understand these concepts:

VXLAN

VXLAN provides the same Ethernet Layer 2 network services as VLAN does today, but with greater extensibility and flexibility. It is a Layer 2 overlay scheme over a Layer 3 network. It uses MAC Address-in-User Datagram Protocol (MAC-in-UDP) encapsulation to provide a means to extend Layer 2 segments across the core network. VXLAN is a solution to support a flexible, large-scale multitenant environment over a shared common physical infrastructure. The transport protocol over the core network is IP plus UDP. Compared to VLAN, VXLAN offers the following benefits:

- Flexible placement of multitenant segments throughout the data center: It provides a solution to extend Layer 2 segments over the underlying shared network infrastructure so that tenant workload can be placed across physical pods in the data center.
- Higher scalability to address more Layer 2 segments: VLANs use a 12-bit VLAN ID to address Layer 2 segments, which results in limiting scalability of only 4094 VLANs. VXLAN uses a 24-bit segment ID known as the VXLAN network identifier (VNID), which enables up to 16 million VXLAN segments to co-exist in the same administrative domain.
- Better utilization of available network paths in the underlying infrastructure: VLAN uses the Spanning Tree Protocol for loop prevention, which ends up not using half of the network links in a network by blocking redundant paths. In contrast, VXLAN packets are transferred through the underlying network based on its Layer 3 header and can take complete advantage of Layer 3 routing, equal-cost multipath (ECMP) routing, and link aggregation protocols to use all available paths.

VXLAN Anycast Gateway

The VXLAN anycast gateway feature extends anycast functionality to VXLAN. It enables the use of anycast routing on a network for underlay multicast load-balancing and redundancy.

The VXLAN anycast solution:

- Allows true active-active first hop gateways (active-active on a per flow basis).
- Does not involve any new control or management plane protocols or any form of external SDN controllers or NMS to co-ordinate and synchronize gateways.

The anycast gateway feature follows these basic concepts:

- Creating a virtual Layer 3 gateway and a virtual VTEP across multiple VXLAN gateways. These gateways use an identical configuration of an overlay IP address, overlay MAC address, and underlay VTEP IP address.
- Creating a private multicast group between the gateways to use as a data plane mirror for certain types of overlay control packets.

Note The VXLAN anycast gateway feature is supported on only Cisco ASR 9000 High Density 100GE Ethernet line cards.

Recommendations

These are the recommendations that users must consider before configuring VXLAN anycast gateway feature:

- BGP does not work with VXLAN anycast feature within a data center.
- IGP works on the underlay network within a data center.
- BGP and IGP should be used on the WAN side.
- Data center top-of-rack (TOR) switches to use static routes between router customer IP to the anycast gateway.

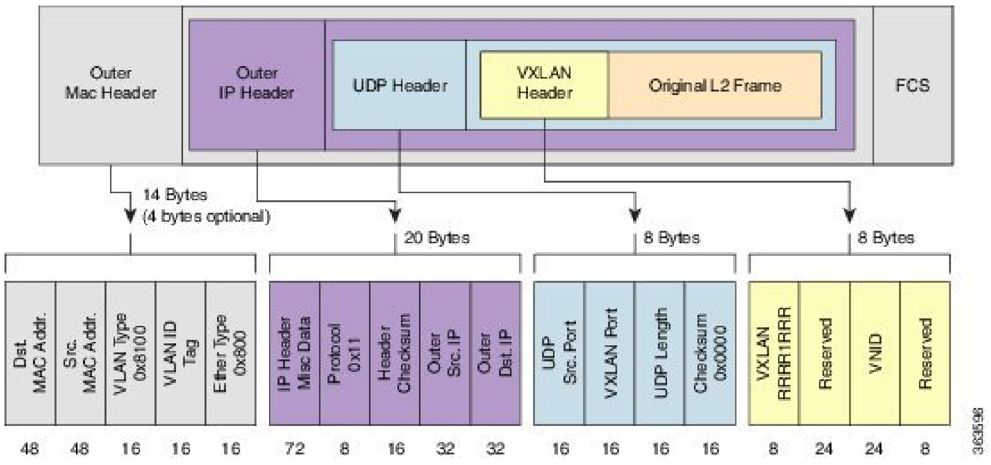
Requirement to deploy VxLAN Anycast Gateway

Since the multicast group is used for mirroring the control frames, in case of IPv6 neighbor advertisements, the duplicate address detection (DAD) protocol will bring down the service because the same addresses are detected between two routers (or interface). Hence you must disable IPv6 DAD on the BVI interface and enable unsolicited node detection (ND) responses.

VXLAN Packet Format

VXLAN defines a MAC-in-UDP encapsulation scheme where the original Layer 2 frame has a VXLAN header added and is then placed in a UDP-IP packet. With this MAC-in-UDP encapsulation, VXLAN tunnels Layer 2 network over Layer 3 network. The VXLAN packet format is shown in the following figure.

Figure 71: VXLAN Packet Format



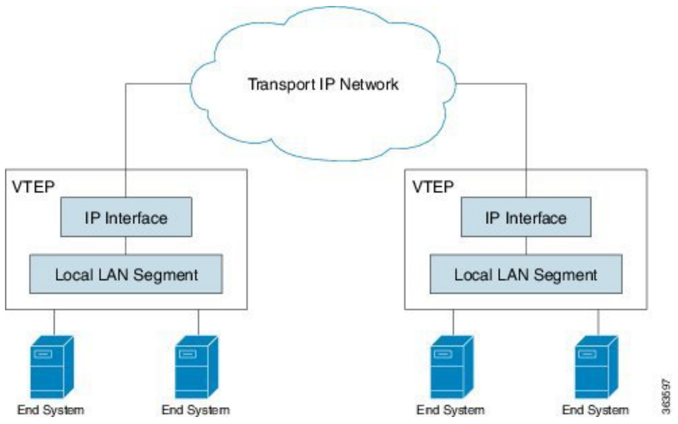
As shown in the above figure, VXLAN introduces an 8-byte VXLAN header that consists of a 24-bit VNID and a few reserved bits. The VXLAN header together with the original Ethernet frame goes in the UDP payload. The 24-bit VNID is used to identify Layer 2 segments and to maintain Layer 2 isolation between the segments. With all 24 bits in VNID, VXLAN can support approx 16 million LAN segments.

VXLAN Tunnel Endpoint

VXLAN uses VXLAN tunnel endpoint (VTEP) devices to map tenants' end devices to VXLAN segments and to perform VXLAN encapsulation and de-encapsulation. Each VTEP function has two interfaces: One is a switch interface on the local LAN segment to support local endpoint communication through bridging, and the other is an IP interface to the transport IP network.

The IP interface has a unique IP address that identifies the VTEP device on the transport IP network known as the infrastructure VLAN. The VTEP device uses this IP address to encapsulate Ethernet frames and transmits the encapsulated packets to the transport network through the IP interface. A VTEP device also discovers the remote VTEPs for its VXLAN segments and learns remote MAC Address-to-VTEP mappings through its IP interface. The functional components of VTEPs and the logical topology that is created for Layer 2 connectivity across the transport IP network is shown in the following figure.

Figure 72: VTEP



The VXLAN segments are independent of the underlying network topology; conversely, the underlying IP network between VTEPs is independent of the VXLAN overlay. It routes the encapsulated packets based on the outer IP address header, which has the initiating VTEP as the source IP address and the terminating VTEP as the destination IP address.

Configuring a Layer 2 VXLAN gateway

A Layer 2 VXLAN gateway bridges traffic between VXLAN and non-VXLAN segments (such as VLAN or VPLS) within the same layer 2 network. The operation of a VXLAN Layer 2 gateway is based on the data plane MAC address learning and flooding of multi-destination traffic such as unknown unicast, multicast, or broadcast frames, using IP multicast. The following sections show how to configure an ASR 9000 series router as a Layer 2 VXLAN gateway between a VLAN and a VXLAN segment in the same L2 domain.

Prerequisites

The following are the prerequisites to configuring a Cisco ASR 9000 series router as a VXLAN Layer 2 gateway:

- Configure a loopback interface. It serves as a source interface for the local VTEP.
- Configure unicast reachability to remote VTEPs.
- Configure Bidirectional Protocol Independent Multicast (Bidir PIM) or PIM Sparse Mode. For more information, see the *Multicast Configuration Guide for Cisco ASR 9000 Series Routers*.

Restrictions

Consider the following restrictions while configuring VXLAN:

- You configure VXLAN only on Overlay Transport Virtualization (OTV) and VXLAN UDP ports.
- The source interface can only be a loopback interface.
- You cannot share a VNI or a multicast group or a source interface across multiple NVE interfaces.
- The VNI range and the multicast range both can only be specified contiguously. A non-contiguous range with comma separated values is not supported.
- The VNI to multicast group mapping can be only either 1:1 or N:1. For example,
  - The "member vni 5000 mcast-group 239.1.1.1" command configures a valid 1:1 mapping.
  - The "member vni 5000-5005 mcast-group 239.1.1.1" command configures a valid N:1 mapping.
- When a VNI is configured as a part of a VNI range, it can be modified or deleted only as part of the same range. For example, if the "member vni 5000-5002 mcast-group 239.1.1.1" command is configured, you cannot disassociate just the VNI 5001 from the NVE interface with a "no member vni 5001" command.
- Static MAC configuration is not supported.
- You can configure a maximum of 128k Layer 2 and Layer 3 sub-interfaces per system. The configuration can be a combination of both Layer 2 sub-interfaces and Layer 3 sub-interfaces; or either fully Layer 2 sub-interfaces or Layer 3 sub-interfaces.

  Though the system allows you to configure more than 128k sub-interfaces per system, you cannot use this configuration for services. Though the system displays a warning message on reaching the threshold of 128k sub-interfaces, the configuration is still applied. However, you cannot use this configuration for services.

Creating and Configuring the Network Virtualization Endpoint (NVE) interface

Perform this task to create an NVE interface and configure it as a VXLAN Tunnel EndPoint (VTEP) for VXLAN.

SUMMARY STEPS

1. **interface nve** *nve-identifier*
2. *(Optional)* **overlay-encapsulation vxlan**
3. **source-interface loopback** *loopback-interface-identifier*
4. **member vni** *vni_number* [*-end_vni_range*] **mcast-group** *ip_address* [*end_ip_address_range*]
5. *(Optional)* **anycast source-interface loopback** *loopback-interface-identifier* **sync-group** *ip_address*
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **interface nve** *nve-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface nve 1
```

Creates the NVE interface and enters the NVE interface configuration sub-mode.

Step 2 *(Optional)* **overlay-encapsulation vxlan**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# overlay-encapsulation vxlan
```

Sets VXLAN encapsulation for the NVE interface. VXLAN is the default encapsulation for an NVE interface. This step is optional if you have not changed the encapsulation.

Step 3 **source-interface loopback** *loopback-interface-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# source-interface loopback 1
```

Sets a loopback interface as the source interface for the VTEP.

Step 4 **member vni** *vni_number* [*-end_vni_range*] **mcast-group** *ip_address* [*end_ip_address_range*]

Example:

```
RP/0/RSP0/CPU0:router(config-if)# member vni 1-10 mcast-group 224.2.2.2 224.2.2.10
```

Associates a single VXLAN or a contiguous range of VXLANs with the NVE interface using their VXLAN Network Identifiers (VNIs) and specifies a multicast address or a contiguous multicast address range associated with these VNIs.

Note
- The mapping between the VNIs and the multicast groups is either one-to-one or many-to-one.
- To associate discontiguous VXLANs or VXLAN ranges with the NVE interface, perform this step for each VXLAN or VXLAN range. For instance,

```
RP/0/RSP0/CPU0:router(config-if)# member vni 10 mcast-group 224.2.2.10
RP/0/RSP0/CPU0:router(config-if)# member vni 23 mcast-group 224.2.2.23
RP/0/RSP0/CPU0:router(config-if)# member vni 50-59 mcast-group 224.2.2.50 224.2.2.59
RP/0/RSP0/CPU0:router(config-if)# member vni 100-120 mcast-group 224.2.2.100 224.2.2.120
```

Step 5 *(Optional)* **anycast source-interface loopback** *loopback-interface-identifier* **sync-group** *ip_address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# anycast source-interface loopback 1 sync-group 192.23.2.20
```

Configures anycast mode parameters for this VTEP.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

What to do next

Use the **show nve interface** command to display the configured NVE interface information.

Creating and configuring a layer 2 sub-interface

Perform this task to create a layer 2 sub-interface associated with a VLAN segment.

SUMMARY STEPS

1. **interface gigabitEthernet** *interface-identifier* **l2transport**
2. **dot1q vlan** *vlan-identifier*
3. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **interface gigabitEthernet** *interface-identifier* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface gigabitEthernet 0/0/0/0.100 l2transport
```

Creates a layer 2 sub-interface and enters the sub-interface configuration mode.

Step 2 **dot1q vlan** *vlan-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# dot1q vlan 100
```

Sets the VLAN for the interface.

Step 3 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Associating VLAN and VXLAN with a bridge domain

Perform this task to associate a VLAN and a VXLAN with a bridge domain.

SUMMARY STEPS

1. **l2vpn**
2. **bridge group** *bridge-group-name*
3. **bridge-domain** *bridge-domain-name*
4. **member vni** *vxlan-identifier*
5. **interface gigabitEthernet** *sub-interface-identifier*
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the l2vpn configuration mode.

Step 2 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bridgegroup1
```

Enters the bridge group configuration mode.

Step 3 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bdomain1
```

Enters the bridge domain configuration mode.

Step 4 **member vni** *vxlan-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# member vni 100
```

Associates a VXLAN with the bridge domain.

Step 5 **interface gigabitEthernet** *sub-interface-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface gigabitEthernet 0/0/0/0.200
```

Associates a VLAN with the bridge domain using the VLAN sub-interface.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring VXLAN source UDP port

This is an optional task. By default, the source UDP port of the encapsulating VXLAN segment is calculated via hash functions on the layer 2 address fields of the inner payload. Perform this task to configure the hash functions to be performed on either the layer 2 or the layer 3 address fields of the inner payload.

SUMMARY STEPS

1. **l2vpn**

2. **load-balancing flow** [*src-dst-mac* | *src-dst-ip*]

DETAILED STEPS

Step 1 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the l2vpn configuration mode.

Step 2 **load-balancing flow** [*src-dst-mac* | *src-dst-ip*]

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# load-balancing flow src-dst-mac
```

Selects either the layer 2 or layer 3 address fields of the inner payload for hash function.

Configuring VXLAN destination UDP port

The UDP port numbers 4789 and 8472 are assigned to VXLAN and OTV respectively. Perform this task to configure the destination UDP port number of the encapsulating VXLAN segment. This is an optional task because, by default, the destination UDP port number of the encapsulating VXLAN datagram is set to 4789. The destination UDP port number should be set to 8472 if the destination VTEP provides VXLAN support using an OTV port.

SUMMARY STEPS

1. **vxlan udp port** *port-number*

DETAILED STEPS

vxlan udp port *port-number*

Example:

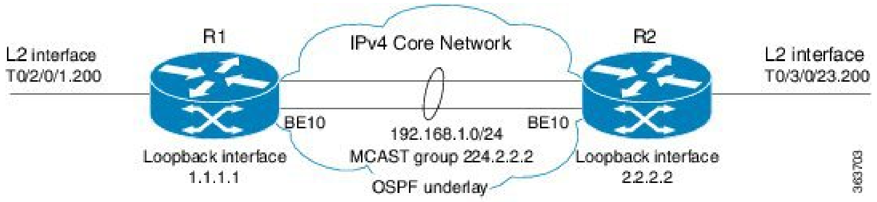
```
RP/0/RSP0/CPU0:router(config)# vxlan udp port 4789
```

Sets the destination UDP port number of the encapsulating VXLAN segment.

Configuration Example for Implementing Layer 2 VXLAN Gateway

The following example shows layer 2 VXLAN gateway configuration on two Provider Edge (PE) routers, R1 and R2, from a sample network topology that has the core network simplified as a bundle link connection between the PE routers.

Figure 73: Network with Layer 2 VXLAN Gateways



Configuration at R1:

```
interface Bundle-Ether10
  ipv4 address 192.168.1.1/24
!
interface Loopback0
  ipv4 address 1.1.1.1/32
!
interface T0/2/0/1
 no shut
!
interface T0/2/0/1.200 l2transport
 encapsulation dot1q 200
!
router ospf underlay
 router-id 1.1.1.1
 area 0
  interface Bundle-Ether10
  interface Loopback0
!
interface nve 1
  member vni 1 mcast-group 224.2.2.2 0.0.0.0
  overlay-encapsulation vxlan
  source-interface Loopback0
!
l2vpn
 bridge group vxlan
  bridge-domain vxlan
    interface T0/2/0/1.200
    member vni 1
   !
multicast-routing
 address-family ipv4
  interface Loopback0
    enable
  interface Bundle-Ether10
     enable
!
router pim
 address-family ipv4
  rp-address 1.1.1.1 bidir
```

Configuration at R2:

```
interface Bundle-Ether10
  ipv4 address 192.168.1.2/24
!
interface Loopback0
  ipv4 address 2.2.2.2/32
!
interface T0/3/0/23
 no shut
!
interface T0/3/0/23.200 l2transport
```

```
 encapsulation dot1q 200
!
router ospf underlay
 router-id 2.2.2.2
 area 0
  interface Bundle-Ether10
  interface Loopback0
!
Interface nve 1
  member vni 1 mcast-group 224.2.2.2 0.0.0.0
  overlay-encapsulation vxlan
  source-interface Loopback0
!
l2vpn
 bridge group vxlan
  bridge-domain vxlan
    interface T0/3/0/23.200
    member vni 1
  !
multicast-routing
 address-family ipv4
  interface Loopback0
    enable
  interface Bundle-Ether10
     enable
!
router pim
 address-family ipv4
  rp-address 1.1.1.1 bidir
```

CHAPTER 11

EVPN Features

This chapter describes how to configure Layer 2 (L2) Ethernet VPN (EVPN) features on the Cisco ASR 9000 Series Aggregation Services Routers supporting Cisco IOS XR software.

EVPN Overview

Ethernet VPN (EVPN) is a next generation solution that provide Ethernet multipoint services over MPLS networks. EVPN operates in contrast to the existing Virtual Private LAN Service (VPLS) by enabling control-plane based MAC learning in the core. In EVPN, PE's participating in the EVPN instances learn customer MAC routes in Control-Plane using MP-BGP protocol. Control-plane MAC learning brings a number of benefits that allow EVPN to address the VPLS shortcomings, including support for multi-homing with per-flow load balancing.

The EVPN control-plane MAC learning has the following benefits:

- Eliminate flood and learn mechanism
- Fast-reroute, resiliency, and faster reconvergence when link to dual-homed server fails
- Enables load balancing of traffic to and from CEs that are multihomed to multiple PEs

The following EVPN modes are supported:

- Single homing - This enables you connect a customer edge (CE) device to one provider edge (PE) device.
- Multihoming - This enables you to connect a customer edge (CE) device to two or more provider edge (PE) devices to provide redundant connectivity. The redundant PE device ensures that there is no traffic disruption when there is a network failure. Following are the types of multihoming:
  - Single-Active - In single-active mode, only a single PE among a group of PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.
  - Active-Active - In active-active mode, all the PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.

EVPN Timers

The following table shows various EVPN timers:

Table 9: EVPN Timers

Timer	Range	Default Value	Trigger	Applicability	Action	Sequence
startup-cost-in	30-86400s	disabled	node recovered*	Single-Homed, All-Active, Single-Active	Postpone EVPN startup procedure and Hold AC link(s) down to prevent CE to PE forwarding. Startup-cost-in timer allows PE to set core protocols first.	1
recovery	20-3600s **Note** Starting from Release 6.6.3 onwards, the range is 0-3600s.	30s	node recovered, interface recovered **	Single-Homed***, Single-Active	Postpone EVPN Startup procedure. Recovery timer allows PE to set access protocols (STP) before reachability towards EVPN core is advertised.	2
peering	0-3600s	3s	node recovered, interface recovered	All-Active, Single-Active	Starts after sending EVPN RT4 to postpone rest of EVPN startup procedure. Peering timer allows remote PE (multihoming AC with same ESI) to process RT4 before DF election will happen.	3

Timer	Range	Default Value	Trigger	Applicability	Action	Sequence
core-de-isolation	60-300s	60s	core interface recovered	Single-Homed[***], Single-Active	Postpone EVPN Startup procedure. Core-de-isolation timer allows EVPN PE nodes to relearn the MAC addresses and BGP routes received from the remote PEs.	

Note

- The timers are available in EVPN global configuration mode and in EVPN interface sub-configuration mode.
- Startup-cost-in is available in EVPN global configuration mode only.
- Timers are triggered in sequence (if applicable).
- Cost-out in EVPN global configuration mode brings down AC link(s) to prepare node for reload or software upgrade.

[*] indicates all required software components are loaded.

[**] indicates link status is up.

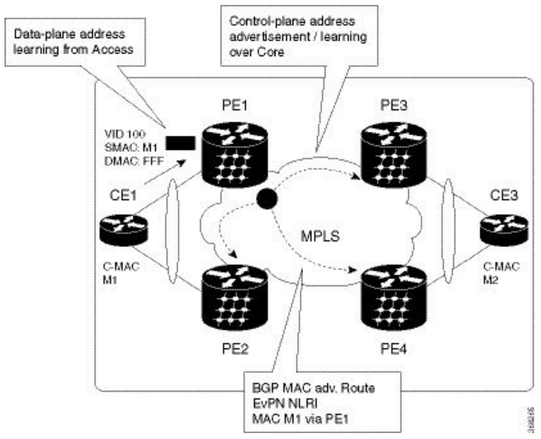
[***] you can change the recovery timer on Single-Homed AC if you do not expect any STP protocol convergence on connected CE.

EVPN Operation

At startup, PEs exchange EVPN routes in order to advertise the following:

- **VPN membership**: The PE discovers all remote PE members of a given EVI. In the case of a multicast ingress replication model, this information is used to build the PE's flood list associated with an EVI.
- **Ethernet segment reachability**: In multi-home scenarios, the PE auto-discovers remote PE and their corresponding redundancy mode (all-active or single-active). In case of segment failures, PEs withdraw ESI-EAD routes and retain EVI-EAD routes used at this stage in order to trigger fast convergence by signaling a MAC mass withdrawal on remote PEs.
- **Redundancy Group membership**: PEs connected to the same Ethernet segment (multi-homing) automatically discover each other and elect a Designated Forwarder (DF) that is responsible for forwarding Broadcast, Unknown unicast and Multicast (BUM) traffic for a given EVI.

Figure 74: EVPN Operation



EVPN can operate in single homing or dual homing mode. Consider single homing scenario, when EVPN is enabled on PE, routes are advertised where each PE discovers all other member PEs for a given EVPN instance. When an unknown unicast (or BUM) MAC is received on the PE, it is advertised as EVPN type-2 routes to other PEs. MAC routes are advertised to the other PEs using EVPN type-2 routes. In multi-homing scenarios Type 1, 3 and 4 are advertised to discover other PEs and their redundancy modes (single active or active-active). Use of Type-1 route is to auto-discover other PE which hosts the same CE. The other use of this route type is to fast route unicast traffic away from a broken link between CE and PE. Type-4 route is used for electing designated forwarder. For instance, consider the topology when customer traffic arrives at the PE, EVPN MAC advertisement routes distribute reachability information over the core for each customer MAC address learned on local Ethernet segments. Each EVPN MAC route announces the customer MAC address and the Ethernet segment associated with the port where the MAC was learned from and is associated MPLS label. This EVPN MPLS label is used later by remote PEs when sending traffic destined to the advertised MAC address.

Behavior Change due to ESI Label Assignment

To adhere to RFC 7432 recommendations, the encoding or decoding of MPLS label is modified for extended community. Earlier, the lower 20 bits of extended community were used to encode the split-horizon group (SHG) label. Now, the SHG label encoding uses from higher 20 bits of extended community.

According to this change, routers in same ethernet-segment running old and new software release versions decodes extended community differently. This change causes inconsistent SHG labels on peering EVPN PE routers. Almost always, the router drops BUM packets with incorrect SHG label. However, in certain conditions, it may cause remote PE to accept such packets and forward to CE potentially causing a loop. One such instance is when label incorrectly read as NULL.

To overcome this problem, Cisco recommends you to:

- Minimize the time both PEs are running different software release versions.
- Before upgrading to a new release, isolate the upgraded node and shutdown the corresponding AC bundle.
- After upgrading both the PEs to the same release, you can bring both into service.

Similar recommendations are applicable to peering PEs with different vendors with SHG label assignment that does not adhere to RFC 7432.

EVPN Route Types

The EVPN network layer reachability information (NLRI) provides different route types.

Table 10: EVPN Route Types

Route Type	Name	Usage
1	Ethernet Auto-Discovery (AD) Route	Few routes sent per ES, carry the list of EVIs that belong to ES
2	MAC/IP Advertisement Route	Advertise MAC, address reachability, advertise IP/MAC binding
3	Inclusive Multicast Ethernet Tag Route	Multicast Tunnel End point discovery
4	Ethernet Segment Route	Redundancy group discovery, DF election

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet (AD) routes are advertised on per EVI and per ESI basis. These routes are sent per ES. They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed.

Route Type 2: MAC/IP Advertisement Route

The host's IP and MAC addresses are advertised to the peers within NRLI. The control plane learning of MAC addresses reduces unknown unicast flooding.

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

Configure EVPN L2 Bridging Service

Perform the following steps to configure EVPN L2 bridging service.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*

4. **bridge-domain** *bridge-domain-name*
5. **interface GigabitEthernet** *GigabitEthernet Interface Instance*
6. **evi** *ethernet vpn id*
7. **exit**
8. **exit**
9. **bridge-domain** *bridge-domain-name*
10. **interface GigabitEthernet** *GigabitEthernet Interface Instance*
11. **evi** *ethernet vpn id*
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the l2vpn configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group 1
```

Enters the bridge group configuration mode.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain 1-1
```

Enters the bridge domain configuration mode.

Step 5 **interface GigabitEthernet** *GigabitEthernet Interface Instance*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
```

Enters interface configuration mode.

Step 6 **evi** *ethernet vpn id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# evi 1
```

Creates the ethernet VPN ID.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac-evi)# exit
```

Exits the current configuration mode.

Step 8 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# exit
```

Exits the current configuration mode.

Step 9 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain 1-2
```

Enters the bridge domain configuration mode.

Step 10 **interface GigabitEthernet** *GigabitEthernet Interface Instance*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# interface GigabitEthernet 0/0/0/1.2
```

Enters interface configuration mode.

Step 11 **evi** *ethernet vpn id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# evi 2
```

Creates the ethernet VPN ID.

Step 12 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

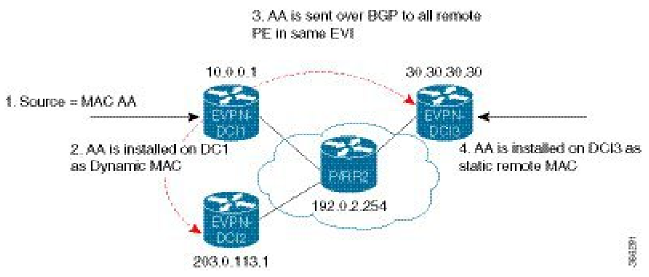
- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

EVPN Software MAC Learning

MAC learning is the method of learning the MAC addresses of all devices available in a VLAN.

The MAC addresses learned on one device needs to be learned or distributed on the other devices in a VLAN. EVPN Native with software MAC Learning feature enables the distribution of the MAC addresses learned on one device to the other devices connected to a network. The MAC addresses are learnt from the remote devices using BGP.

Figure 75: EVPN Native with Software MAC Learning


3. AA is sent over BGP to all remote PE in same EVI
10.0.0.1
30.30.30.30
1. Source = MAC AA
EVPN-DCI1
EVPN-DCI3
2. AA is installed on DCI1 as Dynamic MAC
4. AA is installed on DCI3 as static remote MAC
P/RR2
192.0.2.254
EVPN-DCI2
203.0.113.1

The above figure illustrates the process of Software MAC Learning. The following are the steps involved in the process:

1. Traffic comes in on one port in the bridge domain.
2. The source MAC address (AA) is learnt on DCI1 and is stored as a dynamic MAC entry.
3. The MAC address (AA) is converted into a type-2 BGP route and is sent over BGP to all the remote PEs in the same EVI.
4. The MAC address (AA) is updated on DCI3 as a static remote MAC address.

Software and Hardware Support

The EVPN Native with Software MAC Learning feature is supported on Cisco ASR 9000 Series Routers that support Cisco IOS XR and Cisco IOS XR 64-bit software.

Configure EVPN Native with Software MAC Learning

The following section describes how you can configure EVPN Native with Software MAC Learning:

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_SH
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface TenGigE0/4/0/10.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface BundleEther 20.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# storm-control broadcast pps 10000
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 20.20.20.20 pw-id 1020001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-nbr)# evi 2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# exit
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# exit
RP/0/RSP0/CPU0:router(config-l2vpn)# exit

/* Configure advertisement of MAC routes, suppress unknown unicast, disable the control
word,*/
/* configure the flow label, configure BGP route-exchange using RT. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
/* Use the advertise-mac command to control the advertisement of MAC routes through BGP to
```

```
 other neighbors. */
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
/* Use the unknown-unicast-suppress command to prevent the flooding of unknown unicast
traffic received from the EVPN core towards all other EVPN bridge-ports. */
RP/0/RSP0/CPU0:router(config-evpn-evi)# unknown-unicast-suppress
/* Use the control-word-disable command to prevent the control word from being sent */
/* in the packet that is sent to MPLS core. The control word functionality is enabled by
default. */
RP/0/RSP0/CPU0:router(config-evpn-evi)# control-word-disable
/* Use the load-balance flow label static command to add additional flow label header to
the packet */
/* that is sent to MPLS core. The loadbalance flow functionality is disabled by default.
*/
RP/0/RSP0/CPU0:router(config-evpn-evi)# load-balance flow label static
/* Perform the following steps to configure BGP route-exchange using RT */
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
RP/0/RSP0/CPU0:router(config-evpn-evi)# route-target import 200:101
RP/0/RSP0/CPU0:router(config-evpn-evi)# route-target export 200:101

/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 40.40.40.40
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# description MPLSFACINGPEER
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
```

Supported Modes for EVPN Native with Software MAC Learning

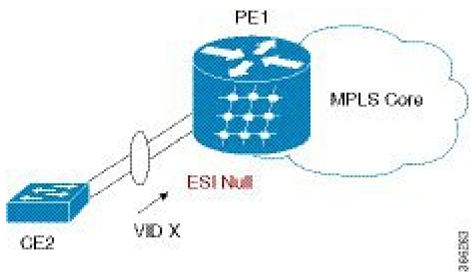
The following are the modes in which EVPN MAC Learning is supported:

- Single Home Device or Single Home Network
- Dual Home Device (DHD) - All Active Load Balancing
- Dual Home Device - Single-Active Load Balancing

Single Home Device or Single Home Network

The following section describes how you can configure EVPN Native with Software MAC Learning feature in single home device or single home network:

Figure 76: Single Home Device or Single Home Network (SHD/SHN)



In the above figure, the PE (PE1) is attached to Ethernet Segment using bundle or physical interfaces. Null Ethernet Segment Identifier (ESI) is used for SHD/SHN.

Configure EVPN in Single Home Device or Single Home Network

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface BundleEther1.2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001

/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac

/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 40.40.40.40
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn
```

Running Configuration

```
l2vpn
bridge group EVPN_ALL_ACTIVE
 bridge-domain EVPN_2001
  interface BundleEther1.2001
  evi 2001
!
evpn
 evi 2001
  advertise-mac
!
router bgp 200 bgp
 router-id 40.40.40.40
 address-family l2vpn evpn
 neighbor 10.10.10.10
  remote-as 200 description MPLS-FACING-PEER
  updatesource Loopback0
  addressfamily l2vpn evpn
```

Verification

Verify EVPN in single home devices.

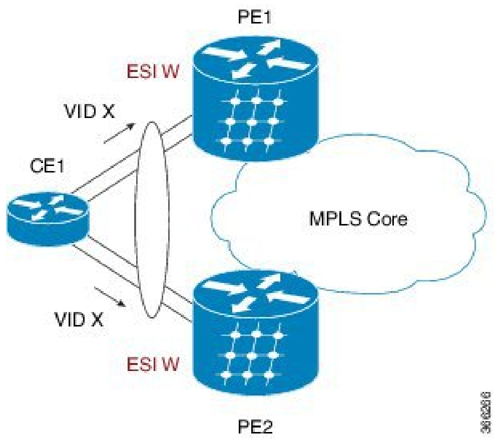
```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface Te0/4/0/10 detail

Ethernet Segment Id    Interface   Nexthops
--------------------   ----------  ----------
N/A                    Te0/4/0/10  20.20.20.20
...............
 Topology :
```

```
Operational : SH
Configured : Single-active (AApS) (default)
```

Dual Home Device—All-Active Load Balancing Mode

The following section describes how you can configure EVPN Software MAC Learning feature in dual home device (DHD) in all-active load balancing mode:

Figure 77: Dual Home Device —All-Active Load Balancing Mode



All-active load-balancing is known as Active/Active per Flow (AApF). In the above figure, identical Ethernet Segment Identifier is used on both EVPN PEs. PEs are attached to Ethernet Segment using bundle interfaces. In the CE, single bundles are configured towards two EVPN PEs. In this mode, the MAC address that is learnt is stored on both PE1 and PE2. Both PE1 and PE2 can forward the traffic within the same EVI.

Configure EVPN Software MAC Learning in Dual Home Device—All-Active Mode

This section describes how you can configure EVPN Software MAC Learning feature in dual home device—all-active mode:

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001


/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
RP/0/RSP0/CPU0:router(config-evpn-evi)# exit
RP/0/RSP0/CPU0:router(config-evpn)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 01.11.00.00.00.00.00.00.01


/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
```

```
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 209.165.200.227
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLS-FACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn


/* Configure Link Aggregation Control Protocol (LACP) bundle. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-if)# lacp switchover suppress-flaps 300
RP/0/RSP0/CPU0:router(config-if)# exit


/* Configure VLAN Header Rewrite.*/

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1 l2transport
RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1q 10
RP/0/RSP0/CPU0:router(config-if)# rewrite ingress tag pop 1 symmetric
```

Note Configure the same mlacp system priority <id> for both the dual homed PE routers to enable all-active load balancing.

Running Configuration

```
l2vpn
bridge group EVPN_ALL_ACTIVE
 bridge-domain EVPN_2001
 interface Bundle-Ether1
 !
 evi 2001
 !
!
evpn
 evi 2001
 !
 advertise-mac
 !
 interface Bundle-Ether1
  ethernet-segment
  identifier type 0 01.11.00.00.00.00.00.00.01
  !
 !
router bgp 200
bgp router-id  209.165.200.227
address-family l2vpn evpn
!
neighbor 10.10.10.10
 remote-as 200
 description MPLS-FACING-PEER
 update-source Loopback0
 address-family l2vpn evpn
!
interface Bundle-Ether1
```

```
lacp switchover suppress-flaps 300
load-interval 30
!
interface Bundle-Ether1 l2transport
 encapsulation dot1aq 2001
 rewrite ingress tag pop 1 symmetric
!
```

Verification

Verify EVPN in dual home devices in All-Active mode.

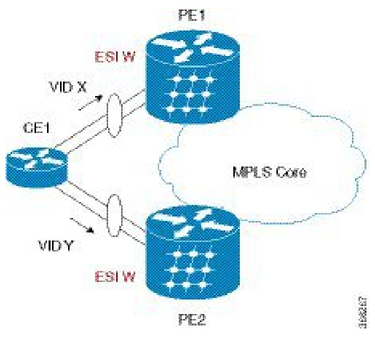
Note With the EVPN IRB, the supported label mode is per-VRF.

```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment interface Bundle-Ether 1 carvin$

Ethernet Segment Id        Interface  Nexthops
-------- ----------        --------   --------
0100.211b.fce5.df00.0b00   BE1       10.10.10.10
 209.165.201.1
Topology :
 Operational : MHN
 Configured : All-active (AApF) (default)
 Primary Services : Auto-selection
 Secondary Services: Auto-selection
 Service Carving Results:
 Forwarders : 4003
 Elected : 2002
 EVI E : 2000, 2002, 36002, 36004, 36006, 36008
 ........
 Not Elected : 2001
 EVI NE : 2001, 36001, 36003, 36005, 36007, 36009

 MAC Flushing mode : Invalid

Peering timer : 3 sec [not running]
 Recovery timer : 30 sec [not running]
 Local SHG label : 34251
 Remote SHG labels : 1
  38216 : nexthop 209.165.201.1
```

Dual Home Device—Single-Active Load Balancing

The following section describes how you can configure EVPN Native with Software MAC Learning feature in dual home device in single-active load balancing mode:

Figure 78: Dual Home Device (DHD)—Single-Active Load Balancing



Single-active load balancing also is known as Active/Active per Service (AApS).

Identical ESI are configured on both EVPN PEs. In the CE, separate bundles or independent physical interfaces are configured towards two EVPN PEs. In this mode, the MAC address that is learnt is stored on both PE1 and PE2. Only one PE can forward traffic within the EVI at a given time.

Configure EVPN in Dual Home Device—Single-Active Mode

```
/* Configure bridge domain. */

RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group EVPN_ALL_ACTIVE
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain EVPN_2001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 2001


/* Configure VLAN Header Rewrite (Single-tagged sub-interface).*/

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1 l2transport
RP/0/RSP0/CPU0:router(config-if)# lacp switchover suppress-flaps 300
RP/0/RSP0/CPU0:router(config-if)# exit
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether1 l2transport
RP/0/RSP0/CPU0:router(config-if)# encapsulation dot1q 10
RP/0/RSP0/CPU0:router(config-if)# rewrite ingress tag pop 1 symmetric


/* Configure advertisement of MAC routes. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac


/* Configure load balancing. */

RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
RP/0/RSP0/CPU0:router(config-evpn-evi)# exit
RP/0/RSP0/CPU0:router(config-evpn)# interface Bundle-Ether1
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# load-balancing-mode single-active
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 12.12.00.00.00.00.00.00.02
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# bgp route-target 1212.0000.0002
```

```
/* Configure address family session in BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# router bgp 200
RP/0/RSP0/CPU0:router#(config-bgp)# bgp router-id 40.40.40.40
RP/0/RSP0/CPU0:router#(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router#(config-bgp)# neighbor 10.10.10.10
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# description MPLSFACING-PEER
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# update-source Loopback 0
RP/0/RSP0/CPU0:router#(config-bgp-nbr)# address-family l2vpn evpn
```

Verification

Verify EVPN in dual home devices in Single-Active mode.

```
RP/0/RSP0/CPU0:router# show evpn ethernet-segment int Bundle-Ether 1 carving detail

...
Ethernet Segment Id       Interface          Nexthops
----- ------------------  ----------         -----------
0012.1200.0000.0000.0002  BE1               10.10.10.10  30.30.30.30

ESI type : 0
 Value : 12.1200.0000.0000.0002
 ES Import RT : 1212.0000.0000 (from ESI)

Source MAC : 0000.0000.0000 (N/A)
Topology :
 Operational : MHN
 Configured : Single-active (AApS)
 Primary Services : Auto-selection
 Secondary Services: Auto-selection

 Service Carving Results:
 Forwarders : 2
 Elected : 1
 EVI E : 500, 2001
 Not Elected : 1
 EVI NE : 501
```

Verify EVPN Native with Software MAC Learning

Verify the packet drop statistics.

```
RP/0/RSP0/CPU0:router# show l2vpn bridge-domain bd-name EVPN_2001 details

Bridge group: EVPN_ALL_ACTIVE, bridge-domain: EVPN_2001, id: 1110,
state: up, ShgId: 0, MSTi: 0
 List of EVPNs:
 EVPN, state: up
 evi: 2001
 XC ID 0x80000458
 Statistics:
 packets: received 28907734874 (unicast 9697466652), sent
76882059953
 bytes: received 5550285095808 (unicast 1861913597184), sent
14799781851396
 MAC move: 0
 List of ACs:
 AC: TenGigE0/4/0/10.2001, state is up
 Type VLAN; Num Ranges: 1
...
 Statistics:
 packets: received 0 (multicast 0, broadcast 0, unknown
unicast 0, unicast 0), sent 45573594908
 bytes: received 0 (multicast 0, broadcast 0, unknown unicast
0, unicast 0), sent 8750130222336
 MAC move: 0
 ........
```

Verify the EVPN EVI information with the VPN-ID and MAC address filter.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 neighbor

Neighbor IP    vpn-id
-----------   --------
20.20.20.20   2001
30.30.30.30   2001
```

Verify the BGP L2VPN EVPN summary.

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn summary
...
Neighbor     Spk   AS     MsgRcvd MsgSent  TblVer    InQ  OutQ  Up/Down  St/PfxRcd
20.20.20.20 0    200     216739  229871   200781341  0    0     3d00h   348032
30.30.30.30 0    200     6462962 4208831  200781341  10   0     2d22h   35750
```

Verify the MAC updates to the L2FIB table in a line card.

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location 0/6/CPU0

Topo ID Producer Next Hop(s)     Mac Address    IP Address
------- -------- -----------     -------------- ----------
1112    0/6/CPU0 Te0/6/0/1.36001 00a3.0001.0001
```

Verify the MAC updates to the L2FIB table in a route switch processor (RSP).

```
RP/0/RSP0/CPU0:router# show l2vpn mac mac all location 0/RSP0/CPU0

Topo ID  Producer     Next Hop(s)     Mac Address    IP Address
-------  --------    -----------     -------------- ----------
1112     0/RSP0/CPU0 Te0/6/0/1.36001 00a3.0001.0001
```

Verify the summary information for the MAC address.

```
RP/0/RSP0/CPU0:router# show l2vpn forwarding bridge-domain EVPN_ALL_ACTIVE:EVPN_2001
mac-address location 0/6/CPU0

.....
Mac Address     Type      Learned from/Filtered on   LC learned   Resync Age/Last Change
Mapped to
0000.2001.5555  dynamic   Te0/0/0/2/0.2001           N/A          11 Jan 14:37:22
N/A <-- local dynamic
00bb.2001.0001 dynamic    Te0/0/0/2/0.2001           N/A          11 Jan 14:37:22
N/A
0000.2001.1111 EVPN       BD id: 1110                N/A          N/A
N/A <-- remote static
```

```
00a9.2002.0001 EVPN       BD id: 1110                N/A          N/A
N/A
```

Verify the EVPN EVI information with the VPN-ID and MAC address filter.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac

EVI    MAC address     IP address        Nexthop      Label
----   -------------   -----------       -------      ------
2001   00a9.2002.0001  ::                10.10.10.10  34226      <-- Remote MAC
2001   00a9.2002.0001  ::                30.30.30.30  34202

2001   0000.2001.5555  20.1.5.55         TenGigE0/0/0/2/0.2001 34203    <-- local MAC


RP/0/RSP0/CPU0:router# RP/0/RSP0/CPU0:router# show evpn evi vpn-id 2001 mac 00a9.2002.0001
 detail

EVI     MAC address      IP address  Nexthop      Label
----    --------------   ----------  -------      -----
2001    00a9.2002.0001   ::          10.10.10.10  34226

2001    00a9.2002.0001   ::          30.30.30.30  34202

 Ethernet Tag : 0
 Multi-paths Resolved : True <--- aliasing to two remote PE with All-Active load balancing

 Static : No
 Local Ethernet Segment : N/A
 Remote Ethernet Segment : 0100.211b.fce5.df00.0b00
 Local Sequence Number : N/A
 Remote Sequence Number : 0
 Local Encapsulation : N/A
 Remote Encapsulation : MPLS
```

Verify the BGP routes associated with EVPN with bridge-domain filter.

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn bridge-domain EVPN_2001 route-type 2

*> [2][0][48][00bb.2001.0001][0]/104
                        0.0.0.0            0 i <------ locally learnt MAC
*>i[2][0][48][00a9.2002.00be][0]/104
                        10.10.10.10 100     0 i <----- remotely learnt MAC
* i 30.30.30.30 100 0 i
```

EVPN Software MAC Aging

You can configure MAC aging on a bridge domain to set the maximum aging time for learned MAC addresses. Decrease the aging time when you want to move the hosts to allow the bridge to adapt to the changes quickly. However, in an EVPN network, the data plane and control plane are always synchronized. Furthermore, it is desirable to have a longer aging times for:

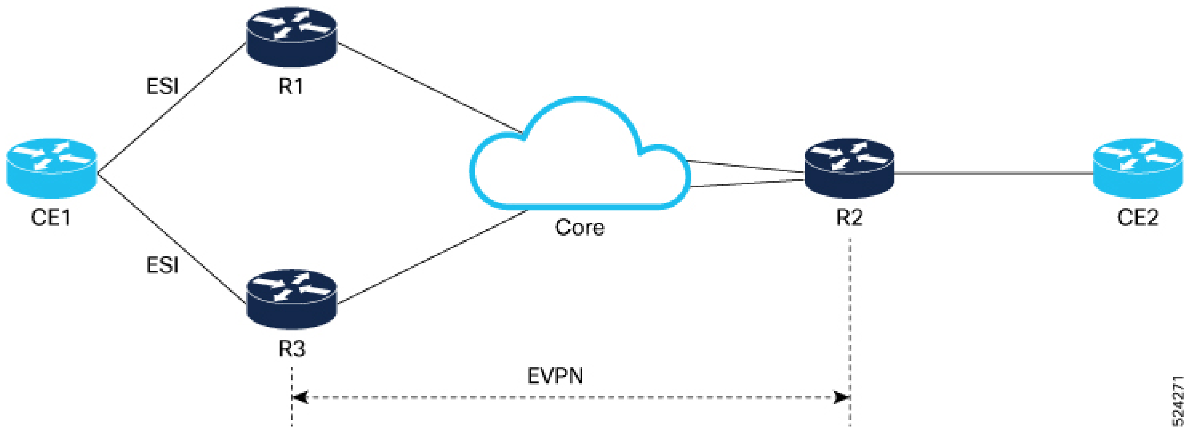
- MAC route stability and reliability
- Support for very high scale of MAC routes

- Reliable and consistent accounting without overloading the control plane

For the above-mentioned reasons, when you enable EVPN, maximum MAC aging times are not fully considered for the configured MAC aging values on the bridge domain. Also, it is observed that the aging times can be long, more than 2 hours.

Topology

Figure 79: MAC Aging in EVPN



For example, CE1 sends a frame to CE2 through router R1 and R3 using EVPN routes, and R2 receives the EVPN routes and learns CE1's MAC address from R1 and R3.

EVPN MAC Aging Process

1. **MAC Address Learning Process**

   When a network device sends a frame, network device learns the MAC address and adds the MAC address to the MAC address table. For example,

   - R1 and R3 learn the MAC address of CE1 and add it to their MAC address tables.
   - R1 and R3 advertise the MAC address to the core network using EVPN routes.
   - R2 receives the EVPN routes and learns MAC address of CE1, adding it to its MAC address table.

2. **EVPN Polling Mechanism**

   Every 15 minutes, the Control Plane (CP) polls the Data Plane (DP) to check if the learned MAC address is still in use. If the CP is busy, it may take up to three polling intervals approximately 45-60 minutes before it can inform the DP about the status of the MAC address.

3. **MAC Aging Process**

   a. **Inactive Customer Edge (CE) Device**

      When a CE device stops sending frames, the DP MAC aging timer for that specific MAC address will expire after one hour. Combined with the CP polling mechanism, the effective MAC aging time is approximately two hours by default which is sum of one hour CP and one hour DP aging.

   b. **Inactive MAC Address Removal**

Once the MAC aging timer expires, the inactive MAC address is removed from the MAC address table.

4. **Re-learning MAC Addresses**

   If a frame from a previously aged-out MAC address is received again, the CP checks if the total MAC addresses in the table have reached 2 million.

   - If the MAC address table is full, the CP disables the MAC address learning process.
   - If the MAC address table is not full, the CP enables the MAC address learning process, and the MAC address is re-learned and added back to the table with a reset aging timer. The MAC aging cycle starts again from Step1, on page 533.

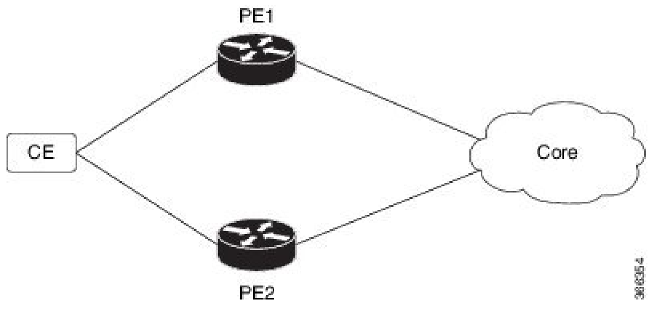
EVPN Out of Service

The EVPN Out of Service feature enables you to control the state of bundle interfaces that are part of an Ethernet segment that have Link Aggregation Control protocol (LACP) configured. This feature enables you to put a node out of service (OOS) without having to manually shutdown all the bundles on their provider edge (PE).

Use the **cost-out** command to bring down all the bundle interfaces belonging to an Ethernet VPN (EVPN) Ethernet segment on a node. The Ethernet A-D Ethernet Segment (ES-EAD) routes are withdrawn before shutting down the bundles. The PE signals to the connected customer edge (CE) device to bring down the corresponding bundle member. This steers away traffic from this PE node without traffic disruption. The traffic that is bound for the Ethernet segment from the CE is directed to the peer PE in a multi-homing environment.

In the following topology, the CE is connected to PE1 and PE2. When you configure the **cost-out** command on PE1, all the bundle interfaces on the Ethernet segment are brought down. Also, the corresponding bundle member is brought down on the CE. Hence, the traffic for this Ethernet segment is now sent to PE2 from the CE.

Figure 80: EVPN Out of Service



To bring up the node into service, use **no cost-out** command. This brings up all the bundle interfaces belonging to EVPN Ethernet segment on the PE and the corresponding bundle members on the CE.

When the node is in cost-out state, adding a new bundle Ethernet segment brings that bundle down. Similarly, removing the bundle Ethernet segment brings that bundle up.

Use **startup-cost-in** command to bring up the node into service after the specified time on reload. The node will cost-out when EVPN is initialized and remain cost-out until the set time. If you execute **evpn no startup-cost-in** command while timer is running, the timer stops and node is cost-in.

The 'cost-out' configuration always takes precedence over the 'startup-cost-in' timer. So, if you reload with both the configurations, cost-out state is controlled by the 'cost-out' configuration and the timer is not relevant. Similarly, if you reload with the startup timer, and configure 'cost-out' while timer is running, the timer is stopped and OOS state is controlled only by the 'cost-out' configuration.

If you do a proc restart while the startup-cost-in timer is running, the node remains in cost-out state and the timer restarts.

Restrictions

- EVPN cost-out is supported only on manually configured ESIs.

Configure EVPN Out of Service

This section describes how you can configure EVPN Out of Service.

```
/* Configuring node cost-out on a PE */

Router# configure
Router(config)# evpn
Router(config-evpn)# cost-out
Router(config-evpn)commit

/* Bringing up the node into service */

Router# configure
Router(config)# evpn
Router(config-evpn)# no cost-out
Router(config-evpn)commit

/* Configuring the timer to bring up the node into service after the specified time on
reload */

Router# configure
Router(config)# evpn
Router(config-evpn)# startup-cost-in 6000
Router(config-evpn)commit
```

Running Configuration

```
configure
evpn
 cost-out
!

configure
evpn
 startup-cost-in 6000
!
```

Verification

Verify the EVPN Out of Service configuration.

```
/* Verify the node cost-out configuration */

Router# **show evpn summary**
Fri Apr  7 07:45:22.311 IST
Global Information
-----------------------------
Number of EVIs               : 2
Number of Local EAD Entries   : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 0
Number of Local MAC Routes    : 5
          MAC                : 5
          MAC-IPv4           : 0
          MAC-IPv6           : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes   : 7
          MAC                : 7
          MAC-IPv4           : 0
          MAC-IPv6           : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels     : 5
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
BGP Router ID                 : ::
BGP ASN                       : 100
PBB BSA MAC address           : 0207.1fee.be00
Global peering timer          :      3 seconds
Global recovery timer         :     30 seconds
EVPN cost-out                 : TRUE
      startup-cost-in timer   : Not configured


/* Verify the no cost-out configuration */

Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
-----------------------------
Number of EVIs               : 2
Number of Local EAD Entries   : 0
Number of Remote EAD Entries  : 0
Number of Local MAC Routes    : 0
Number of Local MAC Routes    : 5
          MAC                : 5
          MAC-IPv4           : 0
          MAC-IPv6           : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes   : 7
          MAC                : 7
          MAC-IPv4           : 0
          MAC-IPv6           : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels     : 5
Number of ES Entries          : 9
Number of Neighbor Entries    : 1
EVPN Router ID                : 192.168.0.1
```

```
BGP Router ID                : ::
BGP ASN                      : 100
PBB BSA MAC address          : 0207.1fee.be00
Global peering timer         :      3 seconds
Global recovery timer        :     30 seconds
EVPN cost-out                : FALSE
      startup-cost-in timer  : Not configured


/* Verify the startup-cost-in timer configuration */

Router# show evpn summary
Fri Apr  7 07:45:22.311 IST
Global Information
-----------------------------
Number of EVIs               : 2
Number of Local EAD Entries  : 0
Number of Remote EAD Entries : 0
Number of Local MAC Routes   : 0
Number of Local MAC Routes   : 5
          MAC                : 5
          MAC-IPv4           : 0
          MAC-IPv6           : 0
Number of Local ES:Global MAC : 12
Number of Remote MAC Routes  : 7
          MAC                : 7
          MAC-IPv4           : 0
          MAC-IPv6           : 0
Number of Local IMCAST Routes : 56
Number of Remote IMCAST Routes: 56
Number of Internal Labels    : 5
Number of ES Entries         : 9
Number of Neighbor Entries   : 1
EVPN Router ID               : 192.168.0.1
BGP Router ID                : ::
BGP ASN                      : 100
PBB BSA MAC address          : 0207.1fee.be00
Global peering timer         :      3 seconds
Global recovery timer        :     30 seconds
EVPN node cost-out           : TRUE
        startup-cost-in timer : 6000
```

EVPN VXLAN Layer 2 Data Center Interconnect Gateway

The Cisco ASR 9000 Series Routers serve as a Data Center Interconnect (DCI) Layer 2 gateway to provide Layer 2 connectivity between EVPN VXLAN based data centers, over a MPLS-based L2VPN network. The data centers are connected through the intermediate service provider network. The EVPN VXLAN enabled data centers use EVPN control plane for distributing Layer 2 forwarding information from one data center to another data center. This feature provides redundancy, resiliency, and ease of provisioning.

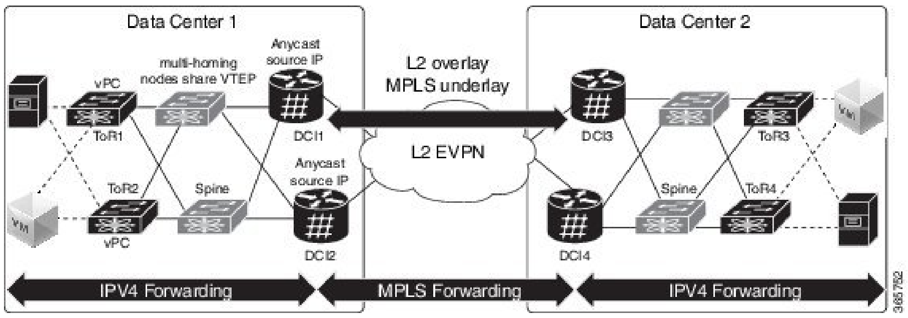
The EVPN VXLAN layer 2 DCI gateway feature supports these functions:

- VXLAN access for single homing
- VXLAN access for all-active multi homing with anycast VXLAN Terminal EndPoint (VTEP) IP address
- VXLAN access for all-active multi homing with unique VTEP IP address
- EVPN ESI Multipath with VXLAN encapsulation

All-Active Multi Homing with Anycast VTEP IP Address

The DCIs use the same anycast VTEP IP address for all-active multi-homing with anycast VTEP IP address. Consider the following topology where Top of Racks (ToRs) are connected to the DCIs using multiple paths: The traffic passes from ToRs to the DCIs through multiple physical paths and uses anycast IP address for load balancing. DCI1 and DCI2 advertise MAC routes to ToRs using the same anycast IP address as that of the next-hop. So, the ToR sends the traffic to the same anycast IP address of the DCIs, and uses IGP ECMP for load balancing. A virtual PortChannel (vPC) allows ToR1 and ToR2 to have the same IP configuration. ToR1 and ToR2 advertise MAC routes to DCIs using the same IP address as that of the next-hop. So, the DCI sends the traffic to the same IP address of the ToRs, and uses IGP ECMP for load balancing. The DCI sends the traffic to the remote data center through MPLS forwarding.

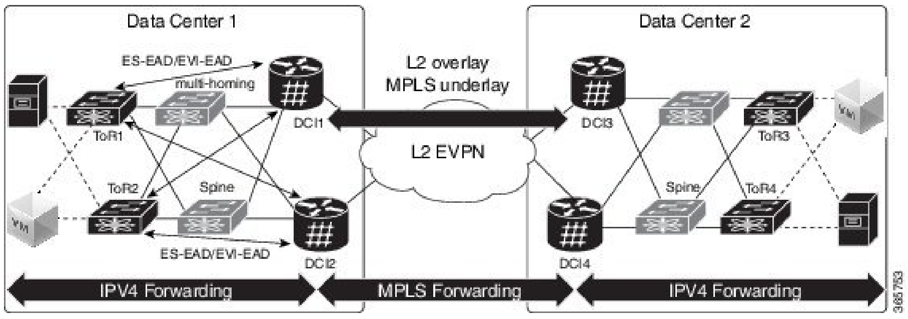
Figure 81: All-Active Multi Homing with Anycast VTEP IP Address



All-Active Multi Homing with Unique VTEP IP Address

The DCIs do not share anycast VTEP IP address for all-active multi homing with unique VTEP IP address. Each DCI uses a unique VTEP IP address. Consider the following topology where ToR receives the MAC routes from DCIs. Each MAC route has a unique next-hop. Because both DCI1 and DCI2 advertise routes for the same MAC with different next-hops, ToR has two equal cost next-hops for the same MAC. When ToR sends the traffic to the MAC, ToR load balances the traffic on both next-hops.

Figure 82: All-Active Multi Homing with Unique VTEP IP Address



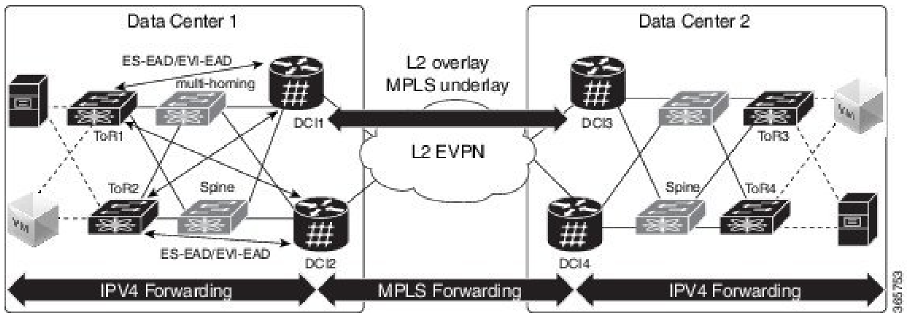
EVPN ESI Multipath for VxLAN - EVI Based Load balancing

The EVPN Ethernet Segment Identifier (ESI) Multipath feature supports multi-path traffic to active-active dual-homed TORs and DCIs to provide redundant connectivity within the data center. ESI multi paths are discovered by the ASR9k DCI router through EVPN signalling. The path selection is based on Ethernet

Segment Identifier (ESI) and EVPN instance (EVI). To resolve paths for MAC routes received, use Ethernet A-D routes per ES (ES-EAD) and Ethernet A-D routes per EVI (EVI-EAD) as specified in RFC 7432.

Consider the following topology where DCIs receive the MAC routes from ToRs and each MAC route has a next-hop for each ToR. Similarly, DCIs advertise MAC routes with different next-hops to ToRs. When DCI sends the traffic to VM, which is behind a pair of ToRs, there are two paths (ToR) for every MAC. The DCI load balances the traffic on the two paths. The selection of path is based on EVI. For example, DCI1 and DCI2 selects ToR1 for all traffic destined to the MAC address learnt on EVI1; DCI1 and DCI2 selects ToR2 for all traffic destined to the MAC address learnt on EVI2.

Figure 83: EVPN ESI Multipath



EVPN ESI Multipath for VxLAN - Flow-based Load Balancing

The EVPN Ethernet Segment Identifier (ESI) Multipath for VxLAN feature supports flow-based load balancing to forward the traffic between Top of Racks (ToRs) and Data Center Interconnect (DCI), and between the source and remote DCIs. A flow is identified either by the source and destination IP address of the traffic, or the source and destination MAC address of the traffic.

In Release 6.2.1, the default load balancing mode is flow-based. You can change the load balancing mode based on per EVI. See Configure Network Virtualization Endpoint (NVE) Interface, on page 546 task to change the load balancing mode based on per EVI.

In Release 6.1.2, only per EVI-based load balancing was supported. Starting from Release 6.2.1, both flow-based load balancing and per EVI based load balancing are supported. The following table shows the support matrix:

Table 11: Support Matrix for EVPN ESI Multipath for VxLAN Load Balancing

Line Card	**Release 6.1.2**	**Release 6.2.1**
ASR 9000 Enhanced Ethernet Line Card	Supports only per EVI-based load balancing	Supports only per EVI-based load balancing

Line Card	Release 6.1.2	Release 6.2.1
A9K-8x100G-LB-SE, A9K-8x100G-LB-TR, A9K-8X100GE-SE, A9K-8X100GE-TR, A9K-4X100GE-SE, A9K-4X100GE-TR, A9K-400G-DWDM-TR, A9K-MOD400-SE, A9K-MOD400-TR, A9K-MOD200-SE, A9K-MOD200-SE	Supports only per EVI-based load balancing	Supports both flow-based and per EVI-based load balancing

The unknown unicast flooding on traffic received from VxLAN segment is supported. In Release 6.2.1, by default, the unknown unicast flooding on traffic received from VxLAN segment is enabled. To disable the unknown unicast flooding, use the **suppress-unknown-unicast-flooding** command. See Configure Network Virtualization Endpoint (NVE) Interface, on page 546 task to disable unknown unicast flooding on traffic received from VxLAN segment.

In Release 6.1.2, by default, the unknown unicast flooding on traffic received from VxLAN segment is disabled.

Table 12: Support Matrix for Unknown Unicast Flooding

Release	Unknown Unicast Flooding
Release 6.1.2	The unknown unicast flooding on traffic received from VxLAN segment is disabled.
Release 6.2.1	The unknown unicast flooding on traffic received from VxLAN segment is enabled. To disable, use the **suppress-unknown-unicast-flooding** command.

Configure EVPN VXLAN Layer 2 Data Center Interconnect Gateway

Perform the following tasks to configure EVPN VXLAN Layer 2 Data Center Interconnect Gateway.

If you want to configure EVPN ESI Multipath feature, do not configure anycast IP address, the remaining configuration tasks remain the same.

Configure L2 EVPN Address Family under BGP Routing Process

Perform this task to enable EVPN address family under BGP routing process.

SUMMARY STEPS

1. **configure**
2. **router bgp** *asn_id*
3. **nsr**
4. **bgp graceful-restart**

5. **bgp router-id** *ip-address*
6. **address-family l2vpn evpn**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **router bgp** *asn_id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# nsr
```

Enables non-stop routing.

Step 4 **bgp graceful-restart**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp graceful-restart
```

Enables graceful restart on the router.

Step 5 **bgp router-id** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 209.165.200.227
```

Configures the router with a specified router ID.

Step 6 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-fmaily l2vpn evpn
```

Enables EVPN address family globally under BGP routing process and enters EVPN address family configuration submode.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.

- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure the Routing Sessions Between the DCI and ToR

Perform this task to configure the routing sessions between the DCI and ToR.

SUMMARY STEPS

1. **configure**
2. **router bgp** *asn_id*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **ebgp-multihop** *maximum hop count*
6. **update-source** *loopback*
7. **address-family l2vpn evpn**
8. **import stitching-rt reoriginate**
9. **route-policy** *route-policy-name* **in**
10. **encapsulation-type** *type*
11. **route-policy** *route-policy-name* **out**
12. **advertise l2vpn evpn re-originated stitching-rt**
13. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **router bgp** *asn_id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 100
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 209.165.200.225
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 209.165.200.225 as a BGP peer.

Step 4 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2000
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **ebgp-multihop** *maximum hop count*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# ebgp-multihop 255
```

Enables multihop peerings with external BGP neighbors.

Step 6 **update-source** *loopback*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source loopback1
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 7 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-fmaily l2vpn evpn
```

Configures EVPN address family.

Step 8 **import stitching-rt reoriginate**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt reoriginate
```

Enables import of routing information from BGP EVPN NLRIs that has route target identifier matching the stitching route target identifier and exports this routing information after re-origination to the L2VPN BGP neighbor.

Step 9 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies the route policy to inbound unicast routes.

Step 10 **encapsulation-type** *type*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
```

Configures VXLAN as encapsulation type.

Step 11 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies the route policy to outbound unicast routes.

Step 12 **advertise l2vpn evpn re-originated stitching-rt**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise l2vpn evpn re-originated stitching-rt
```

Configures advertisement of L2VPN EVPN routes to be received from the L2VPN BGP neighbor.

Step 13 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No**- Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure BGP session for remote DCI Connectivity

Perform this task to configure BGP session for remote DCI connectivity.

SUMMARY STEPS

1. **configure**
2. **router bgp** *asn_id*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **update-source** *loopback*
6. **address-family l2vpn evpn**
7. **import re-originate stitching-rt**
8. **advertise l2vpn evpn re-originated**
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **router bgp** *asn_id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 200
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 209.165.201.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 209.165.201.1 as a BGP peer.

Step 4 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **update-source** *loopback*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source loopback2
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 6 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-fmaily l2vpn evpn
```

Configures EVPN address family.

Step 7 **import re-originate stitching-rt**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
```

Enables import of routing information from BGP EVPN NLRIs that have route target identifier matching the stitching route target identifier, and exports this routing information after re-origination to the L2VPN BGP neighbor.

Step 8 **advertise l2vpn evpn re-originated**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise l2vpn evpn re-originated
```

Configures the advertisement of L2VPN EVPN routes to be received from the L2VPN BGP neighbor.

Step 9 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No**- Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure Network Virtualization Endpoint (NVE) Interface

Perform this task to create an NVE interface and configure it as a VXLAN Tunnel EndPoint (VTEP) for VxLAN.

SUMMARY STEPS

1. **configure**
2. **interface nve** *nve-identifier*
3. **source-interface loopback** *loopback-interface-identifier*
4. **anycast source-interface loopback** *loopback-interface-identifier*
5. **redundancy**
6. **backbone vxlan**
7. **iccp group** *group number*
8. **exit**
9. **backbone mpls**
10. **iccp group** *group number*
11. **exit**
12. **exit**
13. **member vni** *vni_number*
14. **load-balance per-evi**
15. **suppress-unknown-unicast-flooding**
16. **mcast-group** *ip_address*
17. **host-reachability protocol** *protocol*
18. (Optional) **ingress-replication protocol** *protocol*
19. Use the **commit** or **end** command

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **interface nve** *nve-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface nve 1
```

Creates the NVE interface and enters the NVE interface configuration sub-mode.

Step 3 **source-interface loopback** *loopback-interface-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# source-interface loopback 1
```

Sets a loopback interface as the source interface for the VTEP.

Step 4 **anycast source-interface loopback** *loopback-interface-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# anycast source-interface loopback 1
```

Configures anycast mode parameters and source interface for the anycast mode.

Anycast IP address is used for BGP next hop on the fabric side. If you want to configure the ESI multipath feature, do not configure anycast IP address.

Step 5 **redundancy**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# redundancy
```

Configures the redundancy path.

Step 6 **backbone vxlan**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red)# backbone vxlan
```

Configures Inter-Chassis Communication Protocol (ICCP) VXLAN backbone.

Step 7 **iccp group** *group number*

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red-backbone-vxlan)# iccp group 11
```

Configures the ICCP group number.

Step 8 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red-backbone-vxlan)# exit
```

Exits the backbone-vxlan submode and returns to redundancy submode.

Step 9 **backbone mpls**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red)# backbone mpls
```

Configures ICCP MPLS backbone.

Step 10 **iccp group** *group number*

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red-backbone-mpls)# iccp group 12
```

Configures ICCP group number for MPLS backbone.

Step 11 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red-backbone-mpls)# exit
```

Exits the backbone-mpls submode and returns to redundancy submode.

Step 12 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-red)# exit
```

Exits the redundancy submode and returns to interface submode.

Step 13 **member vni** *vni_number*

Example:

```
RP/0/RSP0/CPU0:router(config-nve)# member vni 1
```

Associates a single VxLAN with the NVE interface using the VxLAN Network Identifier (VNI) and specifies a multicast address associated with this VNI.

Step 14 **load-balance per-evi**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-vni)# load-balance per-evi
```

Configures per-evi load balance mode (default is per-flow).

Step 15 **suppress-unknown-unicast-flooding**

Example:

```
RP/0/RSP0/CPU0:router(config-nve-vni)# suppress-unknown-unicast-flooding
```

Configures the suppression of unknown unicast flooding.

Step 16 **mcast-group** *ip_address*

Example:

```
RP/0/RSP0/CPU0:router(config-nve-vni)# mcast-group 209.165.202.129
```

Specifies a multicast address associated with the VNI.

Step 17 **host-reachability protocol** *protocol*

Example:

```
RP/0/RSP0/CPU0:router(config-nve-vni)# host-reachability protocol bgp
```

Configures the BGP control protocol for VxLAN tunnel endpoint reachability.

Step 18 (Optional) **ingress-replication protocol** *protocol*

Example:

```
RP/0/RSP0/CPU0:router(config-nve-vni)# ingress-replication protocol bgp
```

Ingress replication is supported when configured, PIM-SSM otherwise.

Step 19 Use the **commit** or **end** command

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure a Bridge Domain

Perform the following steps to configure the bridge domain on the DCI Gateway.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **evi** *ethernet vpn id*
6. **exit**
7. **member vni** *vxlan-id*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the l2vpn configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
```

Enters the bridge group configuration mode.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
```

Enters the bridge domain configuration mode.

Step 5 **evi** *ethernet vpn id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# evi 1
```

Creates the ethernet VPN ID.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-evi)# exit
```

Exits the EVI configuration mode and returns to bridge domain configuration mode.

Step 7 **member vni** *vxlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# member vni 1
```

Associates a member VNI with the bridge domain.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure BGP Route Targets Import/Export Rules

By default, these parameters are auto-derived from the DCI's configuration:

- Route Distinguisher (RD) for global Ethernet Segment table

Default: Auto-generated RD based on loopback IP address

- EVI's BGP Route Distinguisher (RD)

Default: Auto-generated RD based on loopback IP address

- EVI's BGP Route Target. Default: Auto-generated RT based on EVI ID

Perform this task to overwrite the auto-generated BGP RD/RT values and define route targets to be used for import and export of forwarding information.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **bgp**
4. **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
5. **exit**
6. **evi** *evi_id*

7. **bgp**
8. **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
9. **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
10. **exit**
11. **vni** *vni_id***stitching**
12. **bgp**
13. **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
14. **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# bgp
```

Enters EVPN BGP configuration mode and configures static BGP settings for the Ethernet Segment ES:GLOBAL EVI, which is used for handling ES routes.

Step 4 **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-bgp)# rd 200:50
```

Configures the route distinguisher.

Step 5 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-bgp)# exit
```

Exits the current configuration mode and returns to evpn submode

Step 6 **evi** *evi_id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# evi 1
```

Configures Ethernet VPN ID.

The EVI ID range is from 1 to 65534.

Step 7 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
```

Enters the BGP configuration mode for the specific EVI.

Step 8 **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target import 101:1
```

Configures importing of routes from the L2 EVPN BGP NLRI that have the matching route-target value.

Step 9 **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target export 101:1
```

Configures exporting of routes to the L2 EVPN BGP NLRIs and assigns the specified route-target identifiers to the BGP EVPN NLRIs.

Step 10 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# exit
```

Exits the current configuration mode and returns to evpn submode

Step 11 **vni** *vni_id***stitching**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# vni 1 stitching
```

Configures Ethernet VNI ID. Configures stitching for the VxLAN side.

The VNI ID range is from 1 to 16777215.

Step 12 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-instance)# bgp
```

Enters the BGP configuration mode for the specific VNI.

Step 13 **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-instance-bgp)# route-target import 101:1
```

Configures importing of routes from the L2 EVPN BGP NLRI that have the matching route-target value.

Step 14 **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-instance-bgp)# route-target export 101:1
```

Configures exporting of routes to the L2 EVPN BGP NLRIs and assigns the specified route-target identifiers to the BGP EVPN NLRIs.

Step 15 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure Ethernet Segment Identifier

Perform this task to configure Ethernet Segment Identifier (ESI).

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **interface nve** *nve-identifier*
4. **ethernet-segment**
5. **identifier type** *esi-type esi-identifier*
6. **bgp route-target** *route target value*
7. Use the **commit** or **end** command

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router# evpn
```

Enters EVPN configuration mode.

Step 3 **interface nve** *nve-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# interface nve 1
```

Creates the NVE interface and enters the NVE interface configuration sub-mode

Step 4 **ethernet-segment**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
```

Enters the EVPN ethernet-segment configuration mode.

Step 5 **identifier type** *esi-type esi-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 88.00.00.00.00.00.00.00.01
```

Configures Ethernet Segment Identifier .

Step 6 **bgp route-target** *route target value*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# bgp route-target 8888.0000.0001
```

Configures the BGP import route-target for the Ethernet-Segment.

Step 7 Use the **commit** or **end** command

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure ICCP Group

Perform this task to configure Inter Chassis Communication Protocol (ICCP) parameters.

Configure ICCP group for core interface tracking. If all interfaces are down, the DCI is isolated from the core/fabric network. The associated nve interface is brought down, and BGP NLRIs are withdrawn.

SUMMARY STEPS

1. **configure**
2. **redundancy**
3. **iccp group** *group number*
4. **mode singleton**
5. **backbone**
6. **interface GigabitEthernet** *GigabitEthernet Interface Instance*
7. Use the **commit** or **end** command

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **redundancy**

Example:

```
RP/0/RSP0/CPU0:router(config)# redundancy
```

Enters redundancy configuration mode.

Step 3 **iccp group** *group number*

Example:

```
RP/0/RSP0/CPU0:router(config-redundancy)# iccp group 11
```

Configures ICCP group number.

Step 4 **mode singleton**

Example:

```
RP/0/RSP0/CPU0:router(config-redundancy-iccp-group)# mode singleton
```

Enables to run the group in singleton mode.

Step 5 **backbone**

Example:

```
RP/0/RSP0/CPU0:router(config-redundancy-iccp-group)# backbone
```

Configures ICCP backbone interface.

Step 6 **interface GigabitEthernet** *GigabitEthernet Interface Instance*

Example:

```
RP/0/RSP0/CPU0:router(config-redundancy-group-iccp-backbone)# interface GigabitEthernet 0/2/0/12
```

Configures GigabitEthernet interface.

Step 7 Use the **commit** or **end** command

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Enable Flow-based Load Balancing

Perform this task to enable flow-based load balancing.

SUMMARY STEPS

1. **configure**

2. **l2vpn**
3. **load-balancing flow** *{src-dst-mac | src-dst-ip}*
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the L2VPN configuration mode.

Step 3 **load-balancing flow** *{src-dst-mac | src-dst-ip}*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# load-balancing flow src-dst-ip
```

Enables flow-based load balancing.

Step 4 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Example: All-Active Multi Homing with Anycast VTEP IP Address Configuration

The following example shows the all-active multi homing with anycast VTEP IP address configuration:

```
interface nve1
source-interface loopback1
anycast source-interface loopback2
 member vni 5100
```

```
  mcast-address 239.1.1.1
  host-reachabilty protocol bgp
!

!
evpn
 evi 10
  bgp
   route-target import 100:10
!
 vni 5100 stitching
  bgp
   route-target import 200:5100
   route-target export 200:5100
 !
!
l2vpn
 bridge group DCI
  bridge-domain V1
   evi 10
   member vni 5100
!
router bgp 100
 bgp router-id 209.165.200.226
 address-family l2vpn evpn

!
 neighbor 209.165.201.2
  remote-as 100
  description core-facing
  update-source Loopback1
  address-family l2vpn evpn
   import re-originate stitching-rt
   advertise l2vpn evpn re-originated
!
neighbor 209.165.202.130
  remote-as 200
  ebgp-multihop 255
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy passall in
   encapsulation-type vxlan
   route-policy passall out
   advertise l2vpn evpn re-originated stitching-rt
!
```

Example: All-Active Multi Homing with Unique VTEP IP Address Configuration

The following example shows the all-active multi homing with unique VTEP IP address configuration:

```
interface nve1
source-interface loopback1
 member vni 5100
  mcast-address 239.1.1.1
  host-reachabilty protocol bgp
!
!
```

```
evpn
 evi 10
  bgp
   route-target import 100:10
!
 vni 5100 stitching
  bgp
   route-target import 200:5100
   route-target export 200:5100
 !
!
l2vpn
 bridge group DCI
  bridge-domain V1
   evi 10
   member vni 5100
!
router bgp 100
 bgp router-id 209.165.200.226
 address-family l2vpn evpn

!
neighbor 209.165.201.2
 remote-as 100
 description core-facing
 update-source Loopback1
 address-family l2vpn evpn
   import re-originate stitching-rt
   multipath
   advertise l2vpn evpn re-originated
!
neighbor 209.165.202.130
  remote-as 200
  ebgp-multihop 255
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
    multipath
     route-policy passall in
     encapsulation-type vxlan
     route-policy passall out
     advertise l2vpn evpn re-originated stitching-rt
!
```

EVPN Port-Active Multihoming

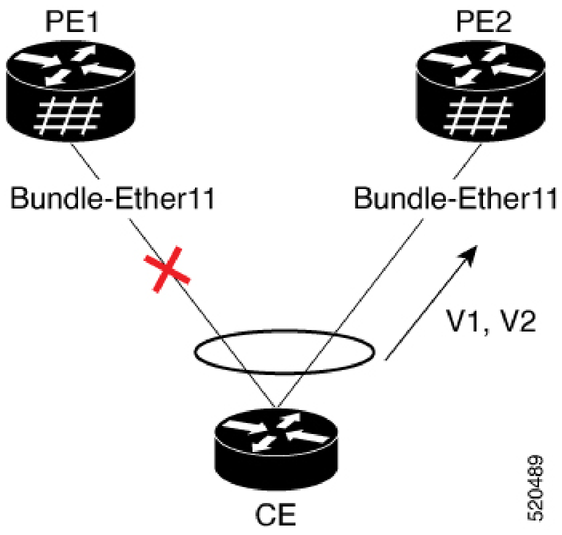
The EVPN Port-Active Multihoming feature supports single-active redundancy load balancing at the port-level or the interface-level. You can use this feature when you want to forward the traffic to a specific interface, rather than have a per-flow load balancing across multiple PE routers. This feature provides a faster convergence during a link failure. This feature enables protocol simplification as only one of the physical ports is active at a given time. You can enable this feature only on bundle interfaces.

EVPN port-active provides protocol simplification compared to Inter-Chassis Communication Protocol (ICCP), which runs on top of Label Distribution Protocol (LDP). You can use this feature as an alternative to multi-chassis link aggregation group (MC-LAG) with ICCP.

Also, you can use this feature when you want certain QoS features to work.

This feature allows one of the PEs to be in active mode and another in the standby mode at the port-level. Only the PE which is in the active mode sends and receives the traffic. The other PE remains in the standby mode. The PEs use the Designated Forwarder (DF) election mechanism to determine which PE must be in the active mode and which must be in the standby mode. You can use either modulo or Highest Random Weight (HRW) algorithm for per port DF election. By default, the modulo algorithm is used for per port DF election.

Figure 84: EVPN Port-Active Multihoming



Consider a topology where the customer edge device (CE) is multihomed to provider edge devices, PE1 and PE2. Use single link aggregation at the CE. Only one of the two interfaces is in the forwarding state, and the other interface is in the standby state. In this topology, PE2 is in the active mode and PE1 is in the standby mode. Hence, PE2 carries traffic from the CE. All services on the PE2 interface operate in the active mode. All services on the PE1 operate in the standby mode.

If the interface is running LACP, then the standby sets the LACP state to Out-of-Service (OOS) instead of bringing the interface state down. This state enables better convergence on standby to active transition.

If you remove the port-active configuration on both PE1 and PE2 and then add back the port-active configuration on both the PEs, PE2 is chosen as an active interface again.

EVPN port-active is compatible with the following services:

- L2 bridging
- L3 gateway
- L2VPN VPLS
- EVPN ELAN
- EVPN IRB
- L2VPN VPWS
- EVPN VPWS
- FXC

Note MC-LAG in EVPN Multihoming is not supported and alternative EVPN port-active should be used.

This feature supports both L2 and L3 port-active functionality. L2 and L3 port-active can coexist on the same bundle. For example, if you configure port-active on a bundle, the bundle can have a mix of both L3 subinterfaces and L2 subinterfaces participating in the services mentioned above.

Configure EVPN Port-Active Multihoming

Perform this task to configure EVPN port-active multihoming.

Configure the same ESI on both the routers. Configure Ethernet-Segment in port-active load-balancing mode on peering PEs for a specific interface.

Configuration Example

```
/* PE1 and PE2 Configuration */

Router#configure
Router(config)#interface Bundle-Ether11
Router(config-if)#lacp system mac 3637.3637.3637
Router(config-if)#exit

Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether11
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 11.11.11.11.11.00.11.11.11
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#commit

/* If you want enable L3 port-active, configure the IP address */
Router#configure
Router(config)#interface Bundle-Ether11
Router(config-if)#ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)#ipv6 address 10::1/64
Router(config-if)#commit
```

Running Configuration

This section shows port-active running configuration.

```
configure
 interface Bundle-Ether11
  lacp system mac 3637.3637.3637
 !

evpn
 interface Bundle-Ether11
  ethernet-segment
   identifier type 0 11.11.11.11.11.00.11.11.11
    load-balancing-mode port-active
   !
  !
interface Bundle-Ether11
 ipv4 address 10.0.0.1 255.0.0.0
 ipv6 address 10::1/64
 !
!
```

Verification

Verify that you have configured the Port-Active Multihoming feature successfully.

```
Router:PE2#show bundle bundle-ether 11

Bundle-Ether11
  Status:                                    Up
  Local links <active/standby/configured>:   1 / 0 / 1
  Local bandwidth <effective/available>:     1000000 (1000000) kbps
  MAC address (source):                      02b4.3cb4.a004 (Chassis pool)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  Load balancing:
    Link order signaling:                    Not configured
    Hash type:                               Default
    Locality threshold:                      None
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured
  IPv6 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------
  Gi0/2/0/8             Local            Active       0x8000, 0x0006     1000000
      Link is Active

/* PE2 is in the active mode, hence the status shows as Up and the Link as Active. */

Router:PE1#show bundle bundle-ether 11

Bundle-Ether11
  Status:                                    LACP OOS  (out of service)
  Local links <active/standby/configured>:   0 / 1 / 1
  Local bandwidth <effective/available>:     0 (0) kbps
  MAC address (source):                      02cf.94c1.0a04 (Chassis pool)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  Load balancing:
    Link order signaling:                    Not configured
    Hash type:                               Default
    Locality threshold:                      None
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured
  IPv6 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------
  Gi0/2/0/7             Local         Standby       0x8000, 0x0006     1000000
```

```
      Link is in standby due to bundle out of service state

/* PE1 is in the standby mode, hence the status shows as LACP OOS  (out of service) and the
 Link is in standby due to bundle out of service state. */

Router:CE#sshow bundle bundle-ether 11

Bundle-Ether11
  Status:                                    Up
  Local links <active/standby/configured>:   1 / 0 / 2
  Local bandwidth <effective/available>:     1000000 (1000000) kbps
  MAC address (source):                      02ff.566c.be04 (Chassis pool)
  Inter-chassis link:                        No
  Minimum active links / bandwidth:          1 / 1 kbps
  Maximum active links:                      64
  Wait while timer:                          2000 ms
  Load balancing:
    Link order signaling:                    Not configured
    Hash type:                               Default
    Locality threshold:                      None
  LACP:                                      Operational
    Flap suppression timer:                  Off
    Cisco extensions:                        Disabled
    Non-revertive:                           Disabled
  mLACP:                                     Not configured
  IPv4 BFD:                                  Not configured
  IPv6 BFD:                                  Not configured

  Port                  Device           State        Port ID         B/W, kbps
  --------------------  ---------------  -----------  --------------
  Gi0/0/0/8             Local            Active       0x8000, 0x0006     1000000
      Link is Active
  Gi0/0/0/16            Local            Negotiating  0x8000, 0x000b     1000000
      Partner is not Synchronized (Waiting, Standby, or LAG ID mismatch)

Router:PE2#show evpn ethernet-segment interface BE11 detail

/* The following output shows that the port-active mode is configured and the port is in
the UP state. */

Ethernet Segment Id             Interface                         Nexthops
------------------------ ---------------------------------- --------------------
0011.1111.1111.0011.1111 BE11                               192.168.0.2
                                                            192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether11
     Interface MAC  : 02b4.3cb4.a004
     IfHandle       : 0x00004170
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.0011.1111
  ES Import RT      : 1111.1111.1100 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
     Mobility-Flush : Count 0, Skip 0, Last n/a
  Peering Details   : 2 Nexthops
```

```
      192.168.0.2 [MOD:P:7fff]
      192.168.0.3 [MOD:P:00]
   Service Carving Results:
      Forwarders     : 0
      Elected        : 0
      Not Elected    : 0
   EVPN-VPWS Service Carving Results:
      Primary        : 0
      Backup         : 0
      Non-DF         : 0
   MAC Flushing mode : STP-TCN
   Peering timer     : 3 sec [not running]
   Recovery timer    : 20 sec [not running]
   Carving timer     : 0 sec [not running]
   Local SHG label   : None
   Remote SHG labels : 0
   Access signal mode: Bundle OOS (Default)


Router:PE1#show evpn ethernet-segment interface BE11 detail

/* The following output shows that the por-active mode is configured and the port is in the
 Standby state. */

Ethernet Segment Id               Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0011.1111.1111.0011.1111 BE11                                  192.168.0.2
                                                               192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether11
     Interface MAC  : 02cf.941c.0a04
     IfHandle       : 0x00004170
     State          : Standby
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.0011.1111
  ES Import RT      : 1111.1111.1100 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
     Mobility-Flush : Count 0, Skip 0, Last n/a
  Peering Details   : 2 Nexthops
     192.168.0.2 [MOD:P:00]
     192.168.0.3 [MOD:P:7fff]
  Service Carving Results:
     Forwarders     : 0
     Elected        : 0
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 20 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : None
  Remote SHG labels : 0
```

```
Access signal mode: Bundle OOS (Default)
```

EVPN Single-Flow-Active Load Multihoming Balancing Mode

Table 13: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Single-Flow-Active Multihoming Load-Balancing Mode	Release 7.3.1	This feature introduces EVPN Single-Flow-Active multihoming mode to connect PE devices in an access network that run Layer 2 access gateway protocols. In this mode, only the PE that first advertises the host MAC address in a VLAN forwards the traffic in a specific flow. When the primary link fails, the traffic quickly switches to the standby PE that learns the MAC address from the originated path, thereby providing fast convergence. A keyword, **single-flow-active** is added to the **load-balancing-mode** command.

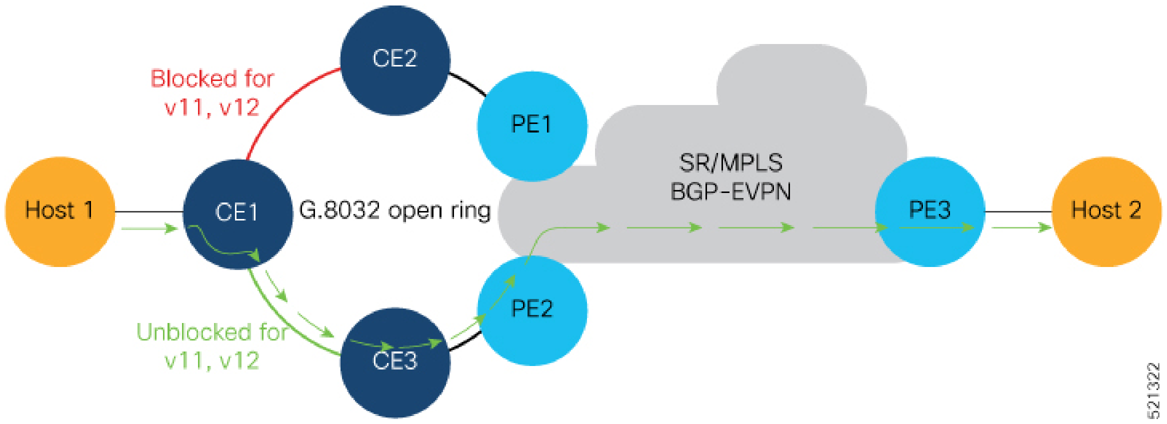
In a ring topology, only one of the PEs, which is the active PE, sends and receives the traffic to prevent a traffic loop. When the link to the active PE fails, the traffic switches over to the standby PE. Traffic switchover takes a while because the standby PE has to learn the MAC addresses of the connected hosts. There's a traffic loss until the traffic switch over happens.

The EVPN Single-Flow-Active multihoming mode connects PE devices in an access network, and in the event of active link failure the switchover happens immediately and reduces the traffic loss.

Both active and standby PEs learn the MAC addresses of the connected host. The PE that learns the MAC address of the host directly is called the Primary (active) PE. The primary PE advertises the learnt MAC addresses to the peer PE, which is referred as standby PE. As the standby PE learns the MAC address of the host through the active PE, this learnt path is referred to as the reoriginated path.

When the primary link fails, the convergence happens fast and the traffic is sent through the standby PE (reoriginated path).

Let us understand how EVPN single flow-active mode helps in fast convergence:



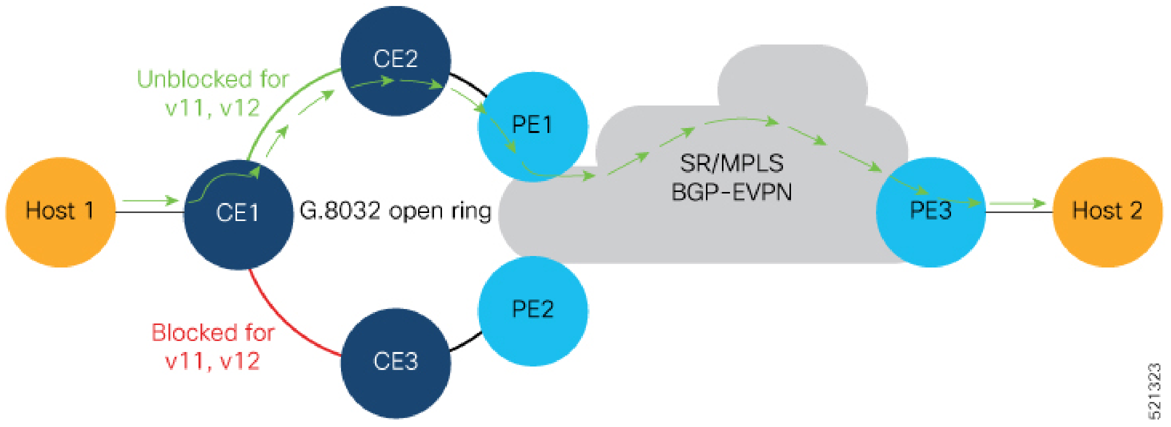
- In this topology, the access network devices are connected through a ring topology. The access network uses Layer-2 gateway protocols such as G.8032, MPLS-TP, STP to prevent traffic loop due to continuous flooding.
- CE1, CE2, CE3, PE1, and PE2 devices form a ring topology.
- Host 1 is connected to CE1.
- CE1 is connected to both PE1 and PE2, thus is multihomed.
- PE1 and PE2 are part of the access ring.
- Both PE1 and PE2 is configured with the same non-zero Ethernet Segment ID (ESI) number `0 36.37.00.00.00.00.00.11.00` for the bundle interface to enable multihoming of the host (CE1).
- PE1 and PE2 belongs to te same VLAN and hence configured with the same EVPN instance (EVI) 100.

Traffic Flow

- Consider a traffic flow from Host 1 to Host 2. The traffic is sent from Host 1 to CE1.
- In this ring topology, the link between CE1 to CE2 is in the blocked state; the link between CE1 to CE3 is in the forwarding state. Hence, CE1 sends the traffic to PE2 through CE3.
- PE2 first learns the MAC address of Host1 through CE1. PE2 advertises the learnt MAC address to the peering PE1.
- As PE2 has learnt the MAC address directly from Host 1, and acts as an active PE.
- The PE which originates the MAC route due to access learning sets the default BGP local preference attribute value to 100.
- PE1 learns the MAC address from PE2 and acts as a stand-by PE. As PE1 gets the reoriginated MAC route from PE2, PE1 sets the BGP local preference attribute value to 80.
- The PE that has the higher local preference always sends and receives the traffic. Thus PE1 sends the traffic to PE3. PE3 sends the traffic to Host 2.

Failure Scenario

When the link between CE1 and CE3 is down or when the link between CE3 and PE2 is down, traffic is sent through PE1.



- When the link fails, the link CE1-CE2 changes to the forwarding state.
- PE1 learns the MAC address of Host 1 directly and advertises the learnt MAC address to PE2.
- PE1 sends the traffic to Host 2 through the remote PE3 with a BGP local preference value of 100.
- PE3 sends and receives the traffic from PE1 until the access link between CE1 and CE2 changes to the blocked state.

Configuration Example

- Configure both PE1 and PE2 with the same EVI of 100.
- Configure both PE1 and PE2 with the same ESI 0 36.37.00.00.00.00.00.11.01.

Perform these tasks on both PE1and PE2.

```
/* Configure advertisement of MAC routes */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# root

/* Configure single-flow-active load-balancing mode */
Router(config)# evpn
Router(config-evpn)# interface bundle-ether 1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 36.37.00.00.00.00.00.11.01
Router(config-evpn-ac-es)# load-balancing-mode single-flow-active
Router(config-evpn-ac-es)# root

/* Configure bridge domain and associating the evi to the bridge domain */
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 100
Router(config-l2vpn-bg)# bridge-domain 100
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1
Router(config-l2vpn-bg-bd-ac)# evi 100
Router(config-l2vpn-bg-bd-ac)# commit
Router(config-l2vpn-bg-bd-evi)# root
Router(config)# exit
Router#
```

Running Configuration

```
evpn
 evi 100
  advertise-mac
  !
 !
 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 36.37.00.00.00.00.00.11.01
   load-balancing-mode single-flow-active
  !
 !
!
l2vpn
 bridge group 100
  bridge-domain 100
   interface Bundle-Ether1
   !
   evi 100
   !
  !
 !
!
```

Verification

Verify the Ethernet Segment Status:

- Verify that the Ethernet Segment Id is the same as that you have configured: In this example, you notice that the ESI on PE1 is 0 36.37.00.00.00.00.00.11.01.
- Verify that the Single-flow-active mode is enabled in the Topology section.

The following example shows the output for PE1:

```
Router# show evpn ethernet-segment carving detail
Thu Aug  6 13:00:37.988 IST
Legend:
  B   - No Forwarders EVPN-enabled,
  C   - Backbone Source MAC missing (PBB-EVPN),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0 36.37.00.00.00.00.00.11.01 BE1                         10.0.0.1
```

```
                                                            172.16.0.1
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 008a.96ee.88dc
     IfHandle       : 0x20005f5c
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 00.0000.0000.0000.0001
  ES Import RT      : 0000.0000.0001 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, Single-flow-active
     Configured     : Single-flow-active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       : MAC-Mobility,
     Mobility-Flush : Debounce 13 sec, Count 1, Skip 1499
                    : Last 01/01 05:57:42.468
  Peering Details   : 2 Nexthops
  10.0.0.1[MOD:P:00:T]
    172.16.0.1 [MOD:P:7fff:T]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
  Service Carving Results:
     Forwarders     : 1000
     Elected        : 1000
            EVI E   :        1,       2,       3,       4,       5,       6
            EVI E   :        7,       8,       9,      10,      11,      12,
            EVI E   :       13,      14,      15,      16,      17,      18,
            EVI E   :       19,      20,      21,      22,      23,      24,
[........]
            EVI E   :      979,     980,     981,     982,     983,     984,
            EVI E   :      985,     986,     987,     988,     989,     990,
            EVI E   :      991,     992,     993,     994,     995,     996,
            EVI E   :      997,     998,     999,    1000
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 29096
  Remote SHG labels : 1
              29096 : nexthop 10.0.0.1
  Access signal mode: Bundle OOS (Default)
```

The following example shows the output for PE2:

```
Router# show evpn ethernet-segment carving detail
Thu Aug  6 13:00:37.988 IST
Legend:
  B   - No Forwarders EVPN-enabled,
  C   - Backbone Source MAC missing (PBB-EVPN),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0 36.37.00.00.00.00.00.11.01 BE1                         10.0.0.1
                                                            172.16.0.1
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 008a.96ee.88dc
     IfHandle       : 0x20005f5c
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 00.0000.0000.0000.0001
  ES Import RT      : 0000.0000.0001 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, Single-flow-active
     Configured     : Single-flow-active
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       : MAC-Mobility,
     Mobility-Flush : Debounce 13 sec, Count 1, Skip 1499
                    : Last 01/01 05:57:42.468
  Peering Details   : 2 Nexthops
  10.0.0.1[MOD:P:00:T]
    172.16.0.1 [MOD:P:7fff:T]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
  Service Carving Results:
     Forwarders     : 1000
     Elected        : 1000
            EVI E   :        1,       2,       3,       4,       5,       6
            EVI E   :        7,       8,       9,      10,      11,      12,
            EVI E   :       13,      14,      15,      16,      17,      18,
            EVI E   :       19,      20,      21,      22,      23,      24,
[........]
            EVI E   :      979,     980,     981,     982,     983,     984,
            EVI E   :      985,     986,     987,     988,     989,     990,
            EVI E   :      991,     992,     993,     994,     995,     996,
            EVI E   :      997,     998,     999,    1000
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 29098
  Remote SHG labels : 1
            29098 : nexthop 172.16.0.1
  Access signal mode: Bundle OOS (Default)
```

Associated Commands

- **load-balancing-mode**
- **show evpn ethernet-segment**

EVPN Convergence Using NTP Synchronization

Table 14: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Convergence Using NTP Synchronization	Release 7.3.1	This feature leverages the NTP clock synchronization mechanism to handle the transfer of DF role from one edge device to another. In this mechanism, the newly added or recovered PE advertises the Service Carving Timestamp along with the current time to peering PEs. This improves convergence by reducing the time for DF election from three seconds to a few tens of milliseconds. The **show evpn ethernet-segment** command is modified to display the Service-Carving wall clock Timestamp (SCT).

In Ethernet VPN, depending on the load-balancing mode, the Designated Forwarder (DF) is responsible for forwarding Unicast, Broadcast, Unknown Unicast, and Multicast (BUM) traffic to a multihomed Customer Edge (CE) device on a given VLAN on a particular Ethernet Segment (ES).

The DF is selected from the set of multihomed edge devices attached to a given ES. When a new edge router joins the peering group either through failure recovery or booting up of a new device, the DF election process is triggered.

By default, the process of transferring the DF role from one edge device to another takes 3 seconds. The traffic may be lost during this period.

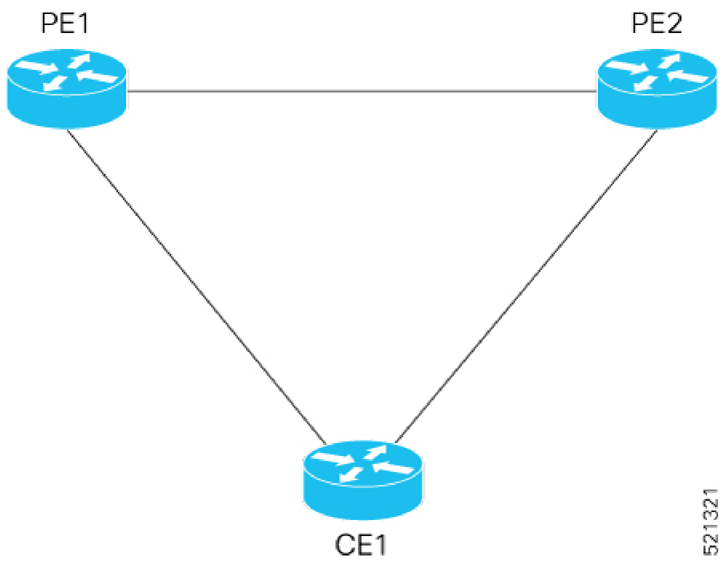
The NTP synchronization mechanism for fast DF election upon recovery leverages the NTP clock synchronization to better align DF events between peering PEs.

If all edge devices attached to a given Ethernet Segment are clock-synchronized with each other using NTP, the default DF election time reduces from 3 seconds to few tens of milliseconds, thereby reducing traffic loss.

Note If the NTP is not synchronized with the NTP server when the EVPN Ethernet Segment interface is coming up, EVPN performs normal DF election.

Let's understand how NTP synchronization works:

Figure 85: EVPN Convergence Using NTP Synchronization



In this topology, CE1 is multihomed to PE1 and PE2.

- PE1 joins the peering group after failure recovery at time (t) = 99 seconds.
- When PE1 joins the peering group, PE1 advertises Route-Type 4 at t = 100 seconds with target Service Carving Time (SCT) value t = 103 seconds to PE2.
- PE2 receives peering Route-Type 4 and learns the DF election time of PE1 to be t =103 seconds.
- If all the peers support NTP, PE2 starts a timer based on the SCT received from PE1 along with a skew value in the Service Carving Time. The skew values are used to eliminate any potential duplicate traffic or loops. Both PE1 and PE2 carves at time t = 103 seconds.

Benefits

- Helps in fast convergence during a primary link recovery
- Supports all the existing load-balancing modes:
  - All-active multihoming
  - Single-active multihoming
  - Port-active multihoming
  - Single-Flow-Active multihoming

Limitations

- All devices attached to a given Ethernet Segment must be configured with NTP. If one of the devices doesn't support NTP clock, the mechanism falls back to default timers.

Verification

Use the **show evpn ethernet-segment** command to view the **Service Carving Time** of the edge device.

For example,

```
Router# show evpn ethernet-segment interface Bundle-Ether200 carving detail

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0053.5353.5353.5353.5301 BE200                              10.0.0.1
                                                            172.16.0.1
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether200
     Interface MAC  : 2c62.34fd.2485
     IfHandle       : 0x20004334
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 53.5353.5353.5353.5301
  ES Import RT      : 8888.8888.8888 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, All-active
     Configured     : All-active (AApF) (default)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       : Reroute
  Peering Details   : 2 Nexthops
     91.0.0.10 [MOD:P:00:T]
     91.0.0.30 [MOD:P:7fff:T]
  Service Carving Synchronization:
     Mode           : NTP_SCT
     Peer Updates   :
               10.0.0.1 [SCT: 2020-10-16 00:28:22:559418]
               10.0.0.3 [SCT: 2020-10-22 17:46:36:587875]
  Service Carving Results:
     Forwarders     : 128
     Elected        : 64

     Not Elected    : 64
```

Associated Commands

- **Show evpn ethernet-segment**

EVPN MPLS Seamless Integration with VPLS

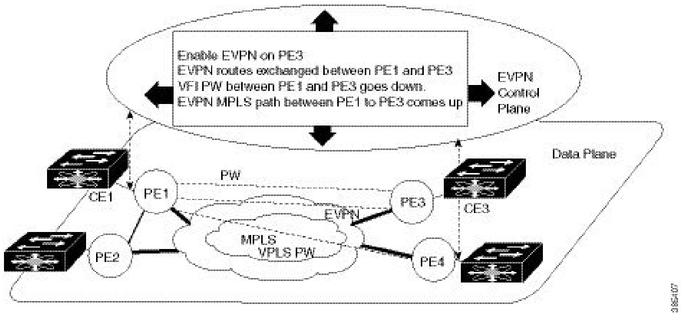
VPLS is a widely-deployed L2VPN technology. As service providers are looking to adopt EVPN on their existing VPLS networks, it is required to provide a mechanism by which EVPN can be introduced without a software upgrade. The EVPN MPLS Seamless Integration with VPLS feature allows EVPN service introduced gradually in the network on a few PE nodes at a time. It eliminates the need to network wide software upgrade at the same time. This feature allows a VPLS service migrated to EVPN service. This feature allows for staged migration where new EVPN sites can be provisioned on existing VPLS enabled PEs. This feature also allows for the co-existence of PE nodes running EVPN and VPLS for the same VPN instance. This allows VPLS or legacy network to be upgraded to the next generation EVPN network without service disruption.

Migrate VPLS Network to EVPN Network through Seamless Integration

In EVPN network, VPN instances are identified by EVPN instance ID (EVI-ID). Similar to other L2VPN technologies, EVPN instances are also associated with route-targets and route-distinguisher. EVPN uses control plane for learning and propagating MAC unlike traditional VPLS, where MAC is learnt in the data plane (learns using "flood and learn technique"). In EVPN, MAC routes are carried by MP-BGP protocol. In EVPN enabled PEs, PEs import the MAC route along with the label to their respective EVPN forwarding table only if their route targets (RTs) match. An EVPN PE router is capable of performing VPLS and EVPN L2 bridging in the same VPN instance. When both EVPN and BGP-AD PW are configured in a VPN instance, the EVPN PEs advertise the BGP VPLS auto-discovery (AD) route as well as the BGP EVPN Inclusive Multicast route (type-3) for a given VPN Instance. Route type-3 referred to as ingress replication multicast route, is used to send broadcast, unknown unicast, and multicast (BUM) traffic. Other remote PEs import type-3 routes for the same VPN instance only if the sending PE RTs match with their configured RT. Thus, at the end of these route-exchanges, EVPN capable PEs discover all other PEs in the VPN instance and their associated capabilities. The type-3 routes used by PE to send its BUM traffic to other PEs ensure that PEs with the same RTs receive the BUM traffic. EVPN advertises the customer MAC address using type-2 route.

This feature allows you to upgrade the VPLS PE routers to EVPN one by one and the network works without any service disruption. Consider the following topology where PE1, PE2, PE3, and PE4 are interconnected in a full-meshed network using VPLS PW.

Figure 86: EVPN MPLS Seamless Integration with VPLS


Enable EVPN on PE3
EVPN routes exchanged between PE1 and PE3
VFI PW between PE1 and PE3 goes down.
EVPN MPLS path between PE1 to PE3 comes up
EVPN Control Plane
Data Plane
PW
CE1
PE1
PE3
CE3
EVPN
MPLS
VPLS PW
PE2
PE4
386407

The EVPN service can be introduced in the network one PE node at a time. The VPLS to EVPN migration starts on PE1 by enabling EVPN in a VPN instance of VPLS service. As soon as EVPN is enabled, PE1 starts advertising EVPN inclusive multicast route to other PE nodes. Since PE1 does not receive any inclusive multicast routes from other PE nodes, VPLS pseudo wires between PE1 and other PE nodes remain up. PE1 keeps forwarding traffic using VPLS pseudo wires. At the same time, PE1 advertises all MAC address learned from CE1 using EVPN route type-2. In the second step, EVPN is enabled in PE3. PE3 starts advertising inclusive multicast route to other PE nodes. Both PE1 and PE3 discover each other through EVPN routes. As a result, PE1 and PE3 shut down the pseudo wires between them. EVPN service replaces VPLS service between PE1 and PE3. At this stage, PE1 keeps running VPLS service with PE2 and PE4. It starts EVPN service with PE3 in the same VPN instance. This is called EVPN seamless integration with VPLS. The VPLS to EVPN migration then continues to remaining PE nodes. In the end, all four PE nodes are enabled with EVPN service. VPLS service is completely replaced with EVPN service in the network. All VPLS pseudo wires are shut down.

Configure EVPN on the Existing VPLS Network

Perform the following tasks to configure EVPN on the existing VPLS network.

- Configure L2VPN EVPN address-family

- Configure EVI and corresponding BGP route-targets under EVPN configuration mode
- Configure EVI under a bridge-domain

See EVI Configuration under L2VPN Bridge-Domain, on page 580 section for how to migrate various VPLS-based network to EVPN.

Configure L2 EVPN Address-Family

Perform this task to enable EVPN address family under both BGP and participating neighbor.

SUMMARY STEPS

1. **configure**
2. **router bgp** *asn_id*
3. **nsr**
4. **bgp graceful-restart**
5. **bgp router-id** *ip-address*
6. **address-family l2vpn evpn**
7. **exit**
8. **neighbor** *ip-address*
9. **remote-as** *autonomous-system-number*
10. **update-source** *loopback*
11. **address-family l2vpn evpn**
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **router bgp** *asn_id*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 65530
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# nsr
```

Enables non-stop routing.

Step 4 **bgp graceful-restart**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp graceful-restart
```

Enables graceful restart on the router.

Step 5 **bgp router-id** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 200.0.1.1
```

Configures the router with a specified router ID.

Step 6 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
```

Enables EVPN address family globally under BGP routing process and enters EVPN address family configuration submode.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# exit
```

Exits the current configuration mode.

Step 8 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 200.0.4.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 200.0.4.1 as a BGP peer.

Step 9 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 65530
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 10 **update-source** *loopback*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback0
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 11 **address-family l2vpn evpn**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
```

Enables EVPN address family globally under BGP routing process and enters EVPN address family configuration submode.

Step 12 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure EVI and Corresponding BGP Route Targets under EVPN Configuration Mode

Perform this task to configure EVI and define the corresponding BGP route targets. Also, configure advertise-mac, else the MAC routes (type-2) are not advertised.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **evi** *evi_id*
4. **bgp**
5. **table-policy** *policy name*
6. **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
7. **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
8. **exit**
9. **advertise-mac**
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 **evi** *evi_id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# evi 1
```

Configures Ethernet VPN ID.

The EVI ID range is from 1 to 65534.

Step 4 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
```

Enters the BGP configuration mode for the specific EVI.

Step 5 **table-policy** *policy name*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# table-policy spp-basic-6
```

Configures policy for installation of forwarding data to L2FIB.

The EVI ID range is from 1 to 65534.

Step 6 **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target import 100:6005
```

Configures importing of routes from the L2 EVPN BGP NLRI that have the matching route-target value.

Step 7 **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target export 100:6005
```

Configures exporting of routes to the L2 EVPN BGP NLRIs and assigns the specified route-target identifiers to the BGP EVPN NLRIs.

Step 8 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# exit
```

Exits the current configuration mode.

Step 9 **advertise-mac**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
```

Advertises MAC route (type-2).

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Example: EVI Configuration under EVPN Configuration-mode

Every participating EVPN instances are identified by EVI_ID. EVI_ID must be defined under EVPN configuration mode as shown below.

```
EVPN
 Evi <VPN ID>
  Bgp
   RD <>
   RT <>
   !
advertise-mac
```

Configure EVI under a Bridge Domain

Perform this task to configure EVI under the corresponding L2VPN bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **interface** *type interface-path-id*
6. **exit**
7. **vfi** { *vfi name* }
8. **neighbor** { *A.B.C.D* } { **pw-id** *value* }
9. **mpls static label local** *label* **remote** *label*
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the L2VPN configuration mode.

Step 3 **bridge group** *bridge group name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
```

Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.

Step 4 **bridge-domain** *bridge-domain name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet0/2/0/0.1
```

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 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit
```

Exits the current configuration mode.

Step 7 **vfi** { *vfi name* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
```

Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

Step 8 **neighbor** { *A.B.C.D* } { **pw-id** *value* }

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000
```

Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the **pw-id** keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

Step 9 **mpls static label local** *label* **remote** *label*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# mpls static label local 20001 remote 10001
```

Configures the MPLS static local label to associate a remote label with a pseudowire or any other bridge interface.

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

EVI Configuration under L2VPN Bridge-Domain

The following examples show EVI configuration under L2VPN bridge-domain for various VPLS-based network:

MPLS static labels based VPLS

```
l2vpn
 bridge group bg1
  bridge-domain bd-1-1
   interface GigabitEthernet0/2/0/0.1
   !
   vfi vfi-1-1
    neighbor 200.0.2.1 pw-id 1200001
     mpls static label local 20001 remote 10001
    !
    neighbor 200.0.3.1 pw-id 1300001
     mpls static label local 30001 remote 10001
    !
    neighbor 200.0.4.1 pw-id 1400001
     mpls static label local 40001 remote 10001
    !
  !
  evi <VPN-ID>
!
```

AutoDiscovery BGP and BGP Signalling based VPLS

```
l2vpn
bridge group bg1
bridge-domain bd-1-2
   interface GigabitEthernet0/2/0/0.2
   !
   vfi vfi-1-2
    vpn-id 2
    autodiscovery bgp
     rd 101:2
     route-target 65530:200
     signaling-protocol bgp
      ve-id 11
      ve-range 16
     !
    !
   evi <VPN-ID>
  !
```

AutoDiscovery BGP and LDP signaling based VPLS

```
l2vpn
bridge group bg1
bridge-domain bd-1-3
   interface GigabitEthernet0/2/0/0.3
   !
   vfi vfi-1-3
    vpn-id 3
    autodiscovery bgp
     rd 101:3
     route-target 65530:300
     signaling-protocol ldp
      vpls-id 65530:3
     !
    !
evi <VPN-ID>
!
```

Targeted LDP based VPLS

```
bridge-domain bd-1-4
   interface GigabitEthernet0/2/0/0.4
   !
   vfi vfi-1-4
    neighbor 200.0.2.1 pw-id 1200004
    !
    neighbor 200.0.3.1 pw-id 1300004
    !
    neighbor 200.0.4.1 pw-id 1400004
    !
   evi <VPN-ID>
   !
```

Verify EVPN Configuration

Verify EVPN configuration and MAC advertisement.

Verify EVPN status, AC status, and VFI status

```
RP/0/#show l2vpn bridge-domain bd-name bd-1-1
Mon Feb 20 21:03:40.244 EST
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: bd-1-1, id: 0, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 1 (1 up), VFIs: 1, PWs: 3 (2 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
    Gi0/2/0/0.1, state: up, Static MAC addresses: 0, MSTi: 2
  List of Access PWs:
  List of VFIs:
    VFI vfi-1-1 (up)
      Neighbor 200.0.2.1 pw-id 1200001, state: up, Static MAC addresses: 0
      Neighbor 200.0.3.1 pw-id 1300001, state: down, Static MAC addresses: 0
      Neighbor 200.0.4.1 pw-id 1400001, state: up, Static MAC addresses: 0
  List of Access VFIs:
 When PEs are evpn enabled, pseudowires that are associated with that BD will be brought
down. The VPLS BD pseudowires are always up.
```

Verify the number of EVI's configured, local and remote MAC-routes that are advertised.

```
RP/0/#show evpn summary
Mon Feb 20 21:05:16.755 EST
-----------------------------
Global Information
-----------------------------
Number of EVIs                    : 6
Number of Local EAD Entries       : 0
Number of Remote EAD Entries      : 0
Number of Local MAC Routes         : 4
          MAC                      : 4
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Local ES:Global MAC      : 1
Number of Remote MAC Routes        : 0
          MAC                      : 0
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Remote SOO MAC Routes    : 0
Number of Local IMCAST Routes      : 4
Number of Remote IMCAST Routes     : 4
Number of Internal Labels          : 0
Number of ES Entries               : 1
Number of Neighbor Entries         : 4
EVPN Router ID                     : 200.0.1.1
BGP ASN                            : 65530
PBB BSA MAC address                : 0026.982b.c1e5
Global peering timer               :      3 seconds
Global recovery timer              :     30 seconds
```

Verify EVPN route-targets.

```
RP/0/#show bgp rt l2vpn evpn
Mon Feb 20 21:06:18.882 EST
EXTCOMM         IMP/EXP
RT:65530:1               1 / 1
RT:65530:2               1 / 1
RT:65530:3               1 / 1
RT:65530:4               1 / 1
Processed 4 entries

Locally learnt MAC routes can be viewed by forwarding table
show l2vpn forwarding bridge-domain mac-address location 0/0/cpu0
To Resynchronize MAC table from the Network Processors, use the command...
    l2vpn resynchronize forwarding mac-address-table location <r/s/i>

Mac Address    Type    Learned from/Filtered on    LC learned Resync Age/Last Change Mapped
 to
-------------- ------- --------------------------- ---------- ----------------------
0033.0000.0001 dynamic Gi0/2/0/0.1                 N/A        20 Feb 21:06:59     N/A

0033.0000.0002 dynamic Gi0/2/0/0.2                 N/A        20 Feb 21:06:59     N/A

0033.0000.0003 dynamic Gi0/2/0/0.3                 N/A        20 Feb 21:04:29     N/A

0033.0000.0004 dynamic Gi0/2/0/0.4                 N/A        20 Feb 21:06:59     N/A

The remote routes learned via evpn enabled BD
show l2vpn forwarding bridge-domain mac-address location 0/0$
To Resynchronize MAC table from the Network Processors, use the command...
    l2vpn resynchronize forwarding mac-address-table location <r/s/i>

Mac Address    Type    Learned from/Filtered on    LC learned Resync Age/Last Change Mapped
 to
-------------- ------- --------------------------- ---------- ----------------------
0033.0000.0001 EVPN    BD id: 0                    N/A        N/A                 N/A

0033.0000.0002 EVPN    BD id: 1                    N/A        N/A                 N/A

0033.0000.0003 EVPN    BD id: 2                    N/A        N/A                 N/A

0033.0000.0004 EVPN    BD id: 3                    N/A        N/A                 N/A
```

Verify EVPN MAC routes pertaining to specific VPN instance.

```
RP/0/#show evpn evi vpn-id 1 mac
Mon Feb 20 21:36:23.574 EST

EVI        MAC address    IP address                   Nexthop
Label
---------- -------------- ---------------------------------------
--------------------------------
1      0033.0000.0001      ::                      200.0.1.1                          45106
```

Verify L2 routing.

```
RP/0/#show l2route evpn mac all
Mon Feb 20 21:39:43.953 EST
Topo ID  Mac Address    Prod   Next Hop(s)
-------- -------------- ------ ----------------------------------------
0        0033.0000.0001 L2VPN  200.0.1.1/45106/ME
1        0033.0000.0002 L2VPN  200.0.1.1/45108/ME
2        0033.0000.0003 L2VPN  200.0.1.1/45110/ME
3        0033.0000.0004 L2VPN  200.0.1.1/45112/ME
```

Verifty EVPN route-type 2 routes.

```
RP/0/#show bgp l2vpn evpn route-type 2
Mon Feb 20 21:43:23.616 EST
BGP router identifier 200.0.3.1, local AS number 65530
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200.0.1.1:1
*>i[2][0][48][0033.0000.0001][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:2
*>i[2][0][48][0033.0000.0002][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:3
*>i[2][0][48][0033.0000.0003][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:4
*>i[2][0][48][0033.0000.0004][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:1 (default for vrf bd-1-1)
*>i[2][0][48][0033.0000.0001][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:2 (default for vrf bd-1-2)
*>i[2][0][48][0033.0000.0002][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:3 (default for vrf bd-1-3)
```

```
*>i[2][0][48][0033.0000.0003][0]/104
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:4 (default for vrf bd-1-4)
*>i[2][0][48][0033.0000.0004][0]/104
                      200.0.1.1                     100      0 i

Processed 8 prefixes, 8 paths
```

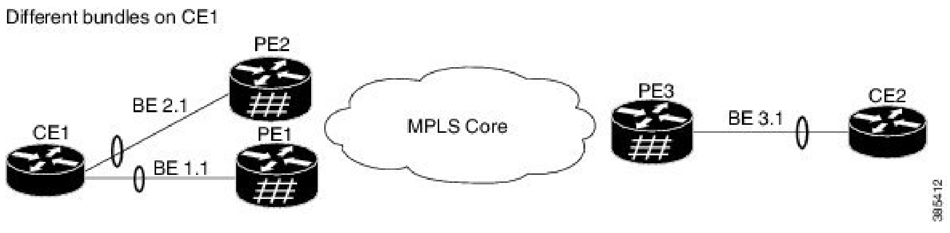
Verify inclusive multicast routes and route-type 3 routes.

```
RP/0/#show bgp l2vpn evpn route-type 3
Mon Feb 20 21:43:33.970 EST
BGP router identifier 200.0.3.1, local AS number 65530
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 21
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 200.0.1.1:1
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:2
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:3
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.1.1:4
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
Route Distinguisher: 200.0.3.1:1 (default for vrf bd-1-1)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
Route Distinguisher: 200.0.3.1:2 (default for vrf bd-1-2)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
Route Distinguisher: 200.0.3.1:3 (default for vrf bd-1-3)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
Route Distinguisher: 200.0.3.1:4 (default for vrf bd-1-4)
*>i[3][0][32][200.0.1.1]/80
                      200.0.1.1                     100      0 i
*> [3][0][32][200.0.3.1]/80
                      0.0.0.0                                0 i
```

EVPN Single-Active Multi-Homing

The EVPN Single-Active Multi-Homing feature supports single-active redundancy mode. In single-active mode, the PE nodes locally connected to an Ethernet Segment load balance traffic to and from the Ethernet Segment based on EVPN service instance (EVI). Within an EVPN service instance, only one PE forwards traffic to and from the Ethernet Segment.

Figure 87: EVPN: Single-Active Multi-Homing



Here is a topology in which CE1 is multihomed to PE1 and PE2. PE1 and PE2 are connected to PE3 through MPLS core. CE3 is connected to PE3 through an Ethernet 'interface bundle. PE1 and PE2 advertise Type 4 routes, and then do designated forwarder (DF) election. The non-DF blocks the traffic in both the directions in single-active mode.

Consider a traffic flow from CE1 to CE2. CE1 sends an address resolution protocol (ARP) broadcast request to both PE1 and PE2. If PE1 is the designated forwarder for the EVI, PE1 forwards the ARP request from CE1. PE2 drops the traffic from CE1. Thereafter, all the unicast traffic is sent through PE1. PE2 will be stand-by or blocked. Traffic is not sent over this path. PE1 advertises MAC to PE3. PE3 always sends and receives traffic through PE1. PE3 sends the traffic to CE2 over Ethernet interface bundle.

Configure EVPN Single-Active Multi-Homing

Perform the following tasks on PE1 and PE2 to configure EVPN Single-Active Multi-Homing feature:

Configuring EVPN Ethernet Segment

Perform this task to configure the EVPN Ethernet segment.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. (Optional) **timers**
4. (Optional) **peering** *seconds*
5. (Optional) **recovery** *seconds*
6. **exit**
7. **interface Bundle-Ether** *bundle-id*
8. **ethernet-segment**
9. **identifier type** *esi-type esi-identifier*
10. **load-balancing-mode single-active**
11. **bgp route-target** *ipv4/v6-address*
12. (Optional) **service-carving manual primary** *{isid}* **secondary** *{isid}*

13. **exit**
14. **exit**
15. (Optional) **mac-flush mvrp**
16. (Optional) **timers**
17. (Optional) **peering** *seconds*
18. (Optional) **recovery** *seconds*
19. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 (Optional) **timers**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# timers
```

Configures global EVPN timers.

Step 4 (Optional) **peering** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-timers)# peering 15
```

Configures the global peering timer. Default is 3 seconds. Range is 0 to 300 seconds.

Step 5 (Optional) **recovery** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-timers)# recovery 30
```

Configures the global recovery timer. Default is 30 seconds. Range is from 20 to 3600 seconds. Starting from Release 6.6.3 onwards, the range is from 0 to 3600 seconds.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-timers)# exit
```

Exits the current configuration mode.

Step 7 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# interface Bundle-Ether1
```

Enters bundle interface configuration mode.

Step 8 **ethernet-segment**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# ethernet-segment
```

Enters the EVPN ethernet-segment configuration mode.

Step 9 **identifier type** *esi-type esi-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
```

Configures the Ethernet segment identifier (ESI) of an interface.

Step 10 **load-balancing-mode single-active**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# load-balancing-mode single-active
```

Specifies the load balancing mode.

Step 11 **bgp route-target** *ipv4/v6-address*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# bgp route-target 4000.0000.0001
```

Configures the BGP Import Route-Target for the Ethernet-Segment.

Step 12 (Optional) **service-carving manual primary** *{isid}* **secondary** *{isid}*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# service-carving manual primary 100 secondary 200
```

Specifies a list of service identifiers (isid) as active and standby services. The isid range is from 256 to 16777216.

Note For ELINE, the isid is the etag. For ELAN, the isid is the EVI. If ELINE and ELAN are used at the same time on a particular ethernet-segment, the isid that matches etag or EVI or both, would apply to carving on ELINE or ELAN or both.

Step 13 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es-man)# exit
```

Exits the current configuration mode.

Step 14 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-es)# exit
```

Exits the current configuration mode.

Step 15 (Optional) **mac-flush mvrp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# mac-flush mvrp
```

Specifies MAC flush mode for this Ethernet Segment.

Step 16 (Optional) **timers**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac)# timers
```

Configures per Ethernet segment timers.

Step 17 (Optional) **peering** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-timers)# peering 15
```

Configures the interface specific peering timer. Default is 3 seconds. Range is 0 to 300 seconds.

Step 18 (Optional) **recovery** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-ac-timers)# recovery 30
```

Configures the interface specific recovery timer. Default is 30 seconds. Range is from 20 to 3600 seconds. Starting from Release 6.6.3 onwards, the range is from 0 to 3600 seconds.

Step 19 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure EVPN Service Instance (EVI) Parameters

Perform this task to define EVPN service instance (EVI) parameters.

SUMMARY STEPS

1. **configure**
2. **evpn**
3. **evi** *evi_id*
4. **bgp**
5. (Optional) **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
6. (Optional) **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
7. (Optional) **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }
8. **exit**
9. **advertise-mac**
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **evpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# evpn
```

Enters EVPN configuration mode.

Step 3 **evi** *evi_id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# evi 6005
```

Configures Ethernet VPN ID.

The EVI ID range is from 1 to 65534.

Step 4 **bgp**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
```

Enters the BGP configuration mode for the specific EVI.

Step 5 (Optional) **rd** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# rd 200:50
```

Configures the route distinguisher.

Step 6 (Optional) **route-target import** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target import 100:6005
```

Configures importing of routes from the L2 EVPN BGP NLRI that have the matching route-target value.

Step 7 (Optional) **route-target export** { *2-byte as_number* | *4-byte as_number* | *IP_address* | **none** } : { *nn* }

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target export 100:6005
```

Configures exporting of routes to the L2 EVPN BGP NLRIs and assigns the specified route-target identifiers to the BGP EVPN NLRIs.

Step 8 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# exit
```

Exits the current configuration mode.

Step 9 **advertise-mac**

Example:

```
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
```

Advertises the MAC route.

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure Layer 2 Interface

Perform this task to define Layer 2 interface.

SUMMARY STEPS

1. **configure**
2. **interface bundle-ether** *instance.subinterface* **l2transport**
3. (Optional) **no shut**
4. **encapsulation dot1q** *vlan-id*
5. (Optional) **rewrite tag pop dot1q** *vlan-id* **symmetric**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **interface bundle-ether** *instance.subinterface* **l2transport**

Example:

```
RP/0/RSP0/CPU0:router(config)# interface bundle-ether2.1 l2transport
```

Configures the bundle ethernet interface and enables Layer 2 transport mode on the bundle ethernet interface.

Step 3 (Optional) **no shut**

Example:

```
RP/0/RSP0/CPU0:router(config-subif-l2)# no shut
```

If a link is in the down state, bring it up. The **no shut** command returns the link to an up or down state depending on the configuration and state of the link.

Step 4 **encapsulation dot1q** *vlan-id*

Example:

```

RP/0/RSP0/CPU0:router(config-subif-l2)# encapsulation dot1q 1
```

Assigns a VLAN attachment circuit to the subinterface.

Step 5 (Optional) **rewrite tag pop dot1q** *vlan-id* **symmetric**

Example:

```
RP/0/RSP0/CPU0:router(config-subif-l2)# rewrite ingress tag pop 1 symmetric
```

Specifies the encapsulation adjustment that is to be performed on the frame ingress to the service instance.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configure a Bridge Domain

Perform the following steps to configure the bridge domain on PE1 and PE2.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge-group-name*
4. **bridge-domain** *bridge-domain-name*
5. **interface Bundle-Ether** *bundle-id*
6. **evi** *ethernet vpn id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the global configuration mode.

Step 2 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the l2vpn configuration mode.

Step 3 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group 6005
```

Enters the bridge group configuration mode.

Step 4 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain 6005
```

Enters the bridge domain configuration mode.

Step 5 **interface Bundle-Ether** *bundle-id*

Example:

```
RP/0/RSP0/CPU0:router(config-evpn)# interface Bundle-Ether2.1
```

Enters bundle interface configuration mode.

Step 6 **evi** *ethernet vpn id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# evi 6005
```

Creates the ethernet VPN ID.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

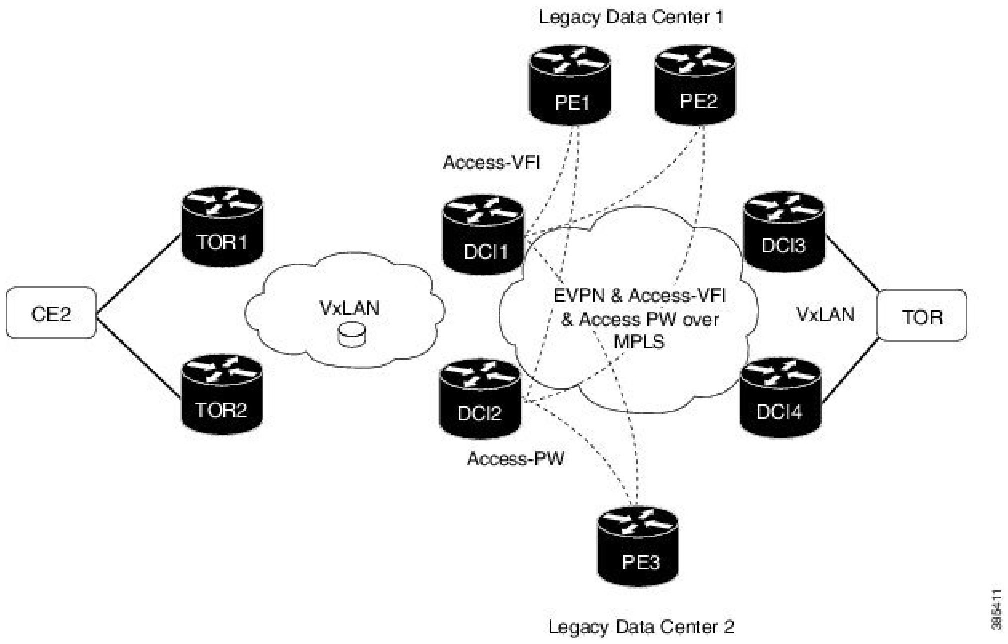
Virtual Ethernet Segment (vES)

Traditionally, multi-homing access to EVPN bridge is through bundle Ethernet connection or a physical Ethernet connection. The Virtual Ethernet Segment (vES) allows a Customer Edge (CE) to access EVPN bridge through MPLS network. The logical connection between CE and EVPN provider edge (PE) is a pseudowire (PW). Using vES you can connect VxLAN EVPN-based data center and a legacy data center through PW based virtual circuit.

The VxLAN EVPN-based data centers and legacy data centers are interconnected through access pseudowire (PW), access virtual forwarding instance (VFI), or both. One vES is created for each access PW and one vES is created per access VFI. This feature supports only single-active mode.

Use access VFI for connecting multiple sites in a mesh topology. Use access PW for connecting few sites in hub and spoke topology.

Figure 88: Virtual Ethernet Segment (vES)



Consider the topology where EVPN data centers are connected to legacy data centers through access PW or access VFI on a single Ethernet segment, which is vES.

Consider a traffic flow from CE2 to PE3. CE2 sends the traffic to DCI1 or DCI2 through EVPN VxLAN. DCI1 and DCI2 are connected to PE3 through access PW on a single Ethernet segment. DCI1 and DCI2 advertise Type 4 routes, and then do designated forwarder (DF) election. The non-DF blocks the traffic on that particular Ethernet segment. Both DCI1 and DCI2 can do the DF election. DCI1 and DCI2 perform DF election after they discover each other. Either one of them can be a DF and other a non-DF. The traffic is forwarded through the DF. The non-DF path is in stand-by mode. DF election is used to prevent traffic loop. DCI1 or DCI2 sends the traffic to PE3.

Consider a traffic flow from CE2 to PE1 and PE2. CE2 sends the traffic to DCI1 or DCI2 through EVPN VxLAN. DCI1 and DCI2 are connected to PE1 and PE2 through access VFI. DCI1 and DCI2 are connected to PE1 and PE2 through access VFI on a single Ethernet segment. DCI1 or DCI2 sends the traffic to PE1 and PE2. DCI1 and DCI2 advertise Type 4 routes, and then do designated forwarder (DF) election. The non-DF blocks the traffic on that particular Ethernet segment. Both DCI1 and DCI2 can do the DF election. DCI1 and DCI2 perform DF election after they discover each other. Either one of them can be a DF and other a non-DF. The traffic is forwarded through the DF. The non-DF path is in stand-by mode. DF election is used to prevent traffic loop. DCI1 or DCI2 sends the traffic to PE3.

Interoperability Between VxLAN and vES

When all-active VxLAN and single-active vES are integrated together, some traffic may take non-optimal path. Consider a traffic flow from CE2 to PE1. VxLAN is in all-active mode and vES is in single active mode. CE2 sends the traffic to ToR1, and ToR1 sends the traffic to DCI1 and DCI2. Both DCI1 and DCI2 can receive the traffic from VxLAN because it is in all-active mode. But, either DCI1 or DCI2 (which is a DF) can forward the traffic through vES. If DCI1 is a non-DF, the traffic is sent from DCI2 to PE1.

Limitations

The vES feature is supported with the following limitations:

- Core isolation is not supported for vES. MPLS core network must be always up and vES redundant peers must be able to exchange type 4 routes while vES is in operation.
- Only targeted LDP pseudowire is supported.
- Interoperability between VxLAN and classic VFI (legacy L2VPN) is not supported.
- Backup PW is not supported with vES.
- PW-status must be supported and enabled on both sides of PW.
- Up to 400 unique RTs are supported for each ESI. However, multiple ESI can share same the RT. Hence, this does not restrict the number of vES.

Configure Virtual Ethernet Segment (vES)

The following sections describe how to configure access PW and access VFI.

Configure Access PW

This section describes how you can configure access PW.

```
/* Configure DCI1 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-bg-bd)# neighbor 70.70.70.70 pw-id 17300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# evi 1
RP/0/RSP0/CPU0:router(config-bg-bd-pw-evi)# member vni 10001

/* Configure EVPN */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# virtual neighbor 70.70.70.70 pw-id 17300001
RP/0/RSP0/CPU0:router(config-evpn-ac-pw)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# identifier type 0 12.12.00.00.00.01.00.00.03
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# bgp route-target 1212.8888.0003
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# exit
RP/0/RSP0/CPU0:router(config-evpn-ac-pw)# timers peering 15
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-timers)# commit
```

```
/* Configure DCI2 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-bg-bd)# neighbor 70.70.70.70 pw-id 27300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# evi 1
RP/0/RSP0/CPU0:router(config-bg-bd-pw-evi)# member vni 10001

/* Configure EVPN */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# virtual neighbor 70.70.70.70 pw-id 27300001
RP/0/RSP0/CPU0:router(config-evpn-ac-pw)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# identifier type 0 12.12.00.00.00.01.00.00.03
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# bgp route-target 1212.8888.0003
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# exit
```

```
RP/0/RSP0/CPU0:router(config-evpn-ac-pw)# timers peering 15
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-timers)# commit
```

```
/* Configure PE3 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group 73
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain 73-1
RP/0/RSP0/CPU0:router(config-bg-bd)# neighbor 10.10.10.10 pw-id 17300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# neighbor 20.20.20.20 pw-id 27300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# commit
```

Running Configuration - Access PW

This section shows access PW running configuration.

```
/* On DCI1 */
!
configure
l2vpn
 bridge group bg1
  bridge-domain bd1
   neighbor 70.70.70.70 pw-id 17300001
    evi 1
     member vni 10001
!

evpn
  virtual neighbor 70.70.70.70 pw-id 17300001
   ethernet-segment
    identifier type 0 12.12.00.00.00.01.00.00.03
    bgp route-target 1212.8888.0003
    !
  timers peering 15
!

/* On DCI2 */
!
configure
l2vpn
 bridge group bg1
  bridge-domain bd1
   neighbor 70.70.70.70 pw-id 27300001
    evi 1
     member vni 10001
!

evpn
  virtual neighbor 70.70.70.70 pw-id 27300001
   ethernet-segment
    identifier type 0 12.12.00.00.00.01.00.00.03
    bgp route-target 1212.8888.0003
    !
  timers peering 15
!

/* On PE3 */
!
configure
l2vpn
 bridge group bg73
  bridge-domain bd73-1
   neighbor 10.10.10.10 pw-id 17300001
   !
   neighbor 20.20.20.20 pw-id 27300001

!
```

Configure Access VFI

This section describes how you can configure access VFI. RTs must match on the redundant DCIs that are connected to the same Ethernet segment.

```
/* Configure DCI1 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-bg-bd)# access-vfi ac-vfi-1
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi)# neighbor 70.70.70.70 pw-id 17100005
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi-pw)# neighbor 80.80.80.80 pw-id 18100005
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi-pw)# exit
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi)# evi 1
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi-evi)# member vni 10001

/* Configure EVPN */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# virtual vfi ac-vfi-1
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# identifier type 0 12.12.00.00.00.01.00.00.01
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# bgp route-target 1212.0005.0001
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# exit
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi)# timers peering 15
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-timers)# exit
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# identifier type 0 12.12.00.00.05.00.00.00.03
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# bgp route-target 1212.0005.0003
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# commit

/* Configure DCI2 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-bg-bd)# access-vfi ac-vfi-1
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi)# neighbor 70.70.70.70 pw-id 27100005
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi-pw)# neighbor 80.80.80.80 pw-id 28100005
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi-pw)# exit
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi)# evi 1
RP/0/RSP0/CPU0:router(config-bg-bd-accessvfi-evi)# member vni 10001

/* Configure EVPN */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# virtual vfi ac-vfi-1
RoRP/0/RSP0/CPU0:routeruter(config-evpn-ac-vfi)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# identifier type 0 12.12.00.00.00.01.00.00.01
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# bgp route-target 1212.0005.0001
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# exit
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi)# timers peering 15
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-timers)# exit
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi)# ethernet-segment
RoRP/0/RSP0/CPU0:routeruter(config-evpn-ac-vfi-es)# identifier type 0
12.12.00.00.05.00.00.00.03
```

```
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# bgp route-target 1212.0005.0003
RP/0/RSP0/CPU0:router(config-evpn-ac-vfi-es)# commit
```

```
/* Configure PE1 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group 71
RoRP/0/RSP0/CPU0:routeruter(config-l2vpn-bg)# bridge-domain 71-1
RP/0/RSP0/CPU0:router(config-bg-bd)# vfi vfi-71-1
RP/0/RSP0/CPU0:router(config-bg-bd-vfi)# neighbor 10.10.10.10 pw-id 17100005
RP/0/RSP0/CPU0:router(config-bg-bd-vfi-pw)# neighbor 20.20.20.20 pw-id 27100005
RP/0/RSP0/CPU0:router(config-bg-bd-vfi-pw)# neighbor 80.80.80.80 pw-id 78100005
RP/0/RSP0/CPU0:router(config-bg-bd-vfi-pw)# commit
```

```
/* Configure PE2 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group 71
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain 71-1
RP/0/RSP0/CPU0:router(config-bg-bd)# vfi vfi-71-1
RP/0/RSP0/CPU0:router(config-bg-bd-vfi)# neighbor 10.10.10.10 pw-id 18100005
RP/0/RSP0/CPU0:router(config-bg-bd-vfi-pw)# neighbor 20.20.20.20 pw-id 28100005
RP/0/RSP0/CPU0:router(config-bg-bd-vfi-pw)# neighbor 70.70.70.70 pw-id 78100005
RP/0/RSP0/CPU0:router(config-bg-bd-vfi-pw)# commit
```

Running Configuration - Access VFI

This section shows access VFI running configuration.

```
/* On DCI1 */
!
configure
l2vpn
 bridge group bg1
  bridge-domain bd1
   access-vfi ac-vfi-1
    neighbor 70.70.70.70 pw-id 17100005
    neighbor 80.80.80.80 pw-id 18100005
   evi 1
    member vni 10001
!
evpn
  virtual vfi ac-vfi-1
   ethernet-segment
    identifier type 0 12.12.00.00.00.01.00.00.01
    bgp route-target 1212.0005.0001
     !
  timers peering 15
!

!

  ethernet-segment
   identifier type 0 12.12.00.00.05.00.00.00.03
   bgp route-target 1212.0005.0003
!




/* On DCI2 */
!
configure
```

```
l2vpn
 bridge group bg1
  bridge-domain bd1
   access-vfi ac-vfi-1
    neighbor 70.70.70.70 pw-id 27100005
    neighbor 80.80.80.80 pw-id 28100005
   evi 1
    member vni 10001
!

evpn
  virtual vfi ac-vfi-1
   ethernet-segment
    identifier type 0 12.12.00.00.00.01.00.00.01
    bgp route-target 1212.0005.0001
    !
  timers peering 15
!
!


  ethernet-segment
   identifier type 0 12.12.00.00.05.00.00.00.03
   bgp route-target 1212.0005.0003
!

/* On PE1 */
!
configure
l2vpn
 bridge group bg71
  bridge-domain bd71-1
   neighbor 10.10.10.10 pw-id 17100005
   !
   neighbor 20.20.20.20 pw-id 27100005
   !
   neighbor 80.80.80.80 pw-id 78100005
!

/* On PE2 */
!
configure
l2vpn
 bridge group bg71
  bridge-domain bd71-1
   neighbor 10.10.10.10 pw-id 18100005
   !
   neighbor 20.20.20.20 pw-id 28100005
   !
   neighbor 70.70.70.70 pw-id 78100005
!
```

AC-based Virtual Ethernet Segment

Table 15: Feature History Table

Feature Name	Release Information	Feature Description

AC-based Virtual Ethernet Segment	Release 7.5.1	This feature allows you to extend the physical links to have VLANs (ACs) that act as Ethernet Virtual Circuits (EVCs). Many such EVCs can be aggregated on a single main interface called Virtual Ethernet Segment (vES). The main interface aggregates many vESs and creates a group to identify these vESs. This mechanism helps to minimize service disruption by mass withdrawal for main peering at the vES level.

Many service providers want to extend the concept of the physical links in an Ethernet Segment. They are looking at having Ethernet Virtual Circuits (EVCs) where many of such EVCs (for example, VLANs) are aggregated on a single physical External Network-to-Network Interface (ENNI). An ES that consists of a set of EVCs instead of physical links is referred to as a virtual ES (vES).

To meet customers' Service Level Agreements (SLA), service providers typically build redundancy through multiple EVPN PEs and across multiple ENNIs where a given vES can be multihomed to two or more EVPN PE devices through their associated EVCs. These Virtual Ethernet Segments (vESes) can be single-homed or multi-homed ES's and when multi-homed, they can operate in either single-active or all-active redundancy modes.

The Ethernet Segment over a parent interface (main port) is represented by parent ES (pES) that can be the main or physical bundle interface. The vES represents the logical connectivity of the access service multi-homed to PE nodes. Multiple vESs are grouped to form one group ES (gES) for one parent interface. This new grouping allows for mass withdrawal of MAC addresses upon main port failure.

The parent interface advertises the grouping ES/EAD (gES/EAD) with the type-3 ESI (meant to represent the main port grouping scheme), which is populated with the six octet MAC address of the main port, and the three octet Local Discriminator value set to 0xFFFFFF.

Similarly, the main port advertises grouping scheme in Type-3 ESI with gES/EAD (and Type-3 ESI also tagged on vES/EAD as an extcomm).

Supported Services

vES supports the following services:

- EVPN ELAN
- EVPN VPWS
- EVPN IRB
- EVPN FXC
- Single-homing load balancing mode
- Multi-homing load balancing mode - active-active and single-active
- Supports Highest Random Weight (HRW) and MODULO algorithm for per port DF election.
- Local switching on the same main port between two vES ACs (ELAN, FXC)

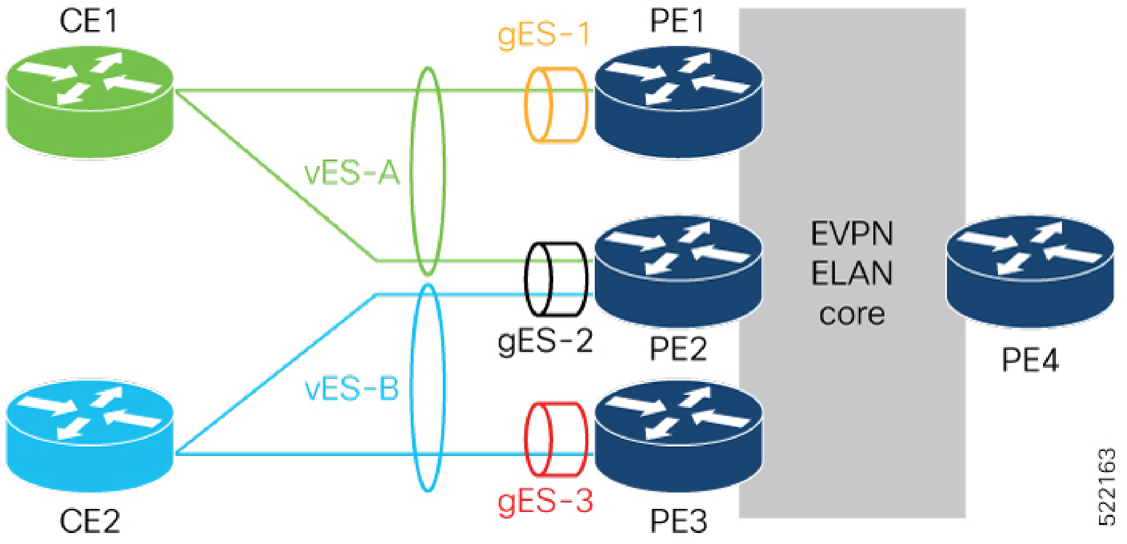
Restrictions

- You might observe a traffic drop during the AC shutdown with vES.

- For vES subinterface, the L3 route-sync is not supported when the main-port is vES-enabled. The syslog or warning message is not reported when the L3 subinterface is configured with VRF **evpn-route-sync**.

Topology

In this example, vES-A is setup between PE1 and PE2. On PE1, there is a grouping ES gES-1 on the access facing interface. Similarly, on PE2 there is also a grouping ES gES-2.



In this topology, the following shows how PEs are peered:

- PE1 and PE2 routers peer using vES-A with RT-4 (each route colored with gES-1 and gES-2 respectively).
- PE2 and PE3 routers peer using vES-B with RT-4 (each route colored with gES-2 and gES-3 respectively).

The following information depicts how traffic is forwarded:

PE4 connects vES-B remotely through PE2 and PE3:

- vES-B - MAC2 [PE3]
- vES-B - EVI/EAD [PE2/L2, PE3/L3)
- vES-B - ES/EAD [PE2 (gES-2), PE3 (gES-3)]
- gES-2 - ES/EAD [PE2]
- gES-3 - ES/EAD [PE3]

PE3 connects vES-A remotely through PE1 and PE2:

- vES-A - MAC1 [PE1]
- vES-A - EVI/EAD [PE1/L1, PE2/L2)
- vES-A - ES/EAD [PE1 (gES-1), PE2 (gES-2)]
- gES-1 - ES/EAD [PE1]
- gES-2 - ES/EAD [PE2]

PE1 performs the same forwarding for PE3 for vES-B.

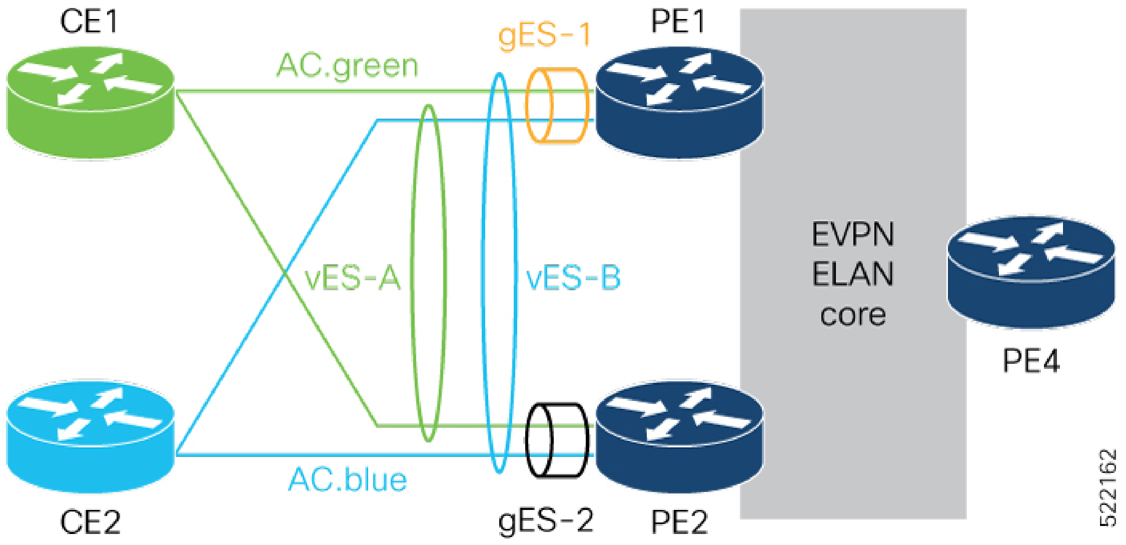
The following routes are advertised with the vESI in the NLRI:

- RT-4 at the granularity of vES for peering and DF-election, along with BGP router MAC extcomm carrying grouping scheme value (gES), which is the main port MAC address. BGP extcomm carries six bytes data which is exactly the length of MAC address.
- Any locally learned MAC address through RT-2 for bridging.
- Per EVI/EAD for service reachability.
- Per ES/EAD for that vES along with BGP router MAC extcomm carrying gES MAC address.

Local Switching

Local switching allows you to switch Layer 2 data between two ACs on the same interface. Local switching involves the exchange of L2 data from one attachment circuit (AC) to the other, and between two interfaces of the same type on the same router. 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.

Consider an example where the customer is provided a service by two different SPs. PE1 and PE2 can local-switch between vES-A and vES-B.



In this topology, the following shows how PEs are peered:

- PE1 and PE2 are peered for vES-A with RT-4
- PE1 and PE2 are peered for vES-B with RT-4

For BUM traffic, traffic is flooded to other ACs in Split-Horizon Group 0.

For Unicast traffic, the MAC lookup in the bridge forwards the traffic to the right AC.

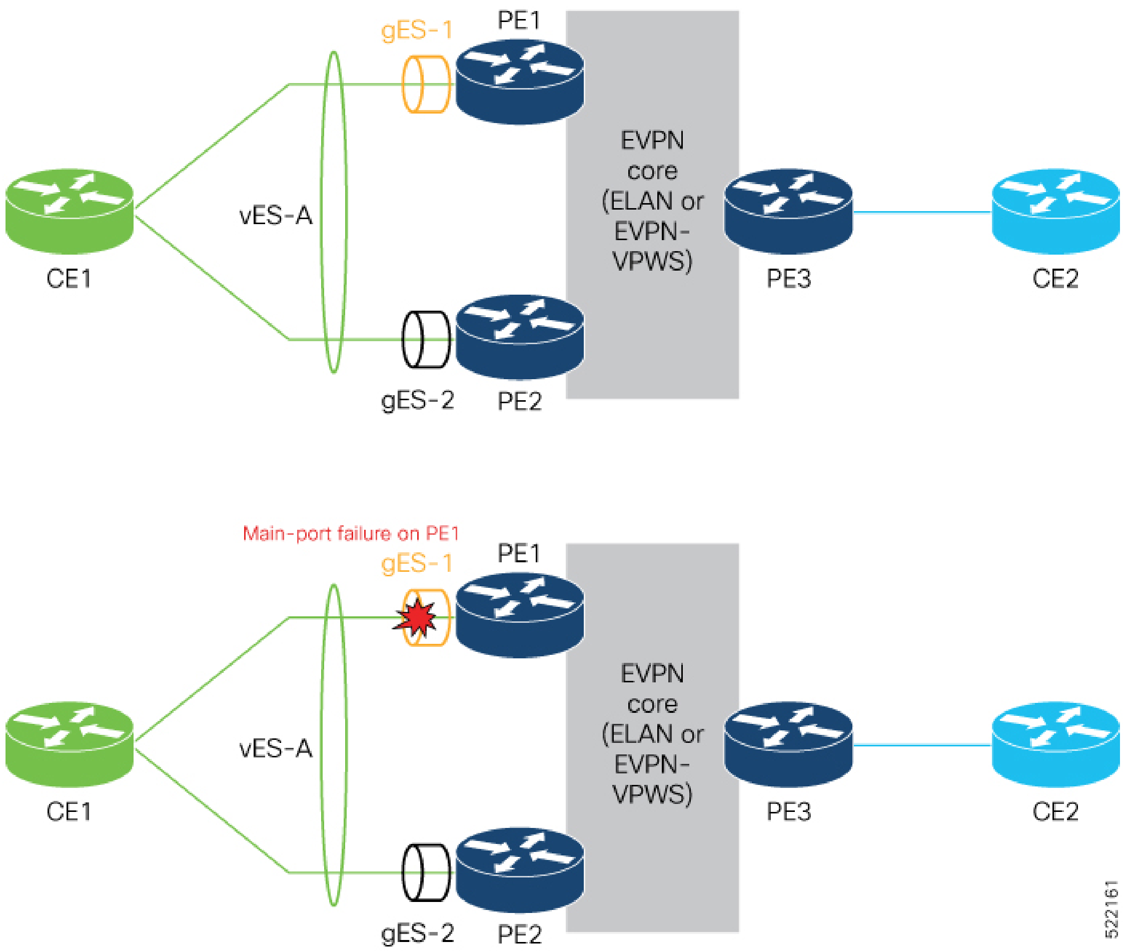
If the local switching is not available, for example the AC goes down, then traffic is routed through the EVPN core. PE1 and PE2 will see each other's remote EVI/EAD and ES/EAD routes for vES-A and vES-B along with pES1 and pES2 ES/EAD.

Main Port Failure

When there is a main port failure, the gES/EAD is withdrawn to provide fast switchover. The vES EVI/EAD and vES/EAD are advertised. After the main port recovery, the gES/EAD is re-advertised on the last vES to prevent remote end steering traffic to node.

The vES failure is identified as an AC failure, and is signaled through CFM/OAM. During vES failure, not the main port failure, the vES EVI/EAD is advertised and the vES/EAD is withdrawn. On vES recovery, after the peering timer expires, the vES/EAD is advertised.

Figure 89:



The following are remote routes for PE3

- vES-A EVI/EAD [PE1/L1,PE2/L2]
- vES-A ES/EAD [PE1 [gESI-1],PE2 [gESI-2]]
- gES-1 ES/EAD [PE1]
- gES-2 ES/EAD [PE2]

After the main port failure, PE3 sees the following remote routes:

- vES-A EVI/EAD [PE1/L1,PE2/L2]

- vES-A ES/EAD [PE1 [gESI-1],PE2 [gESI-2]]
- gES-2 ES/EAD [PE2]
- gES-1 ES/EAD [PE1] is withdrawn

Configure Virtual Ethernet Segment - AC based

Configuration Example

The following example depicts a simple configuration for all-active vES sub-interface and non-vES enabled sub-interface under the same main-port:

```
evpn
  virtual interface Bundle-Ether1.1
    ethernet-segment
      identifier type 0 1.2.3.4.5.6.7.8.9
!
l2vpn  bridge-group g1
    bridge-domain d1
      interface Bundle-Ether1.1      >>MH vES
      interface Bundle-Ether1.2      >>Becomes SH
      evi 1
```

The following example depicts an expanded configuration to flex all options for vES (single-active bundle vES sub-interface with static gES-MAC):

```
  interface Bundle-Ether1
    ethernet-segment
       load-balancing-mode single-active
    virtual-ethernet-segment
       identifier type 3 000a.000b.000c  !
  virtual interface Bundle-Ether1.1
    ethernet-segment
      identifier type 0 1.2.3.4.5.6.7.8.9
!
l2vpn  bridge-group g1
    bridge-domain d1
      interface Bundle-Ether1.1          >>MH vES
      interface Bundle-Ether1.2          >>Becomes SH
      evi 1
```

Verification

Verify the vES AC carving details.

```
Router# show evpn ethernet-segment interface bundle-Ether 5555.1 carving detail

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0055.5555.aabb.0000.0001 BE5555.1                 10.201.201.201
                                                                    10.250.250.250

  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Virtual Access    :
     Interface name : Bundle-Ether5555.1
     IfHandle       : 0x20017b36
```

```
      State          : Up
   **ESI type**          **: 0**
      **Value**          **: 55.5555.aabb.0000.0001**
   ES Import RT      : 6500.1111.2222 (Local)
   Source MAC        : 0000.0000.0000 (N/A)
   Topology          :
      Operational    : MH, All-active
      Configured     : All-active (AApF) (default)
   **Service Carving   : HRW**
      Multicast      : Disabled
   Convergence       :
   Peering Details   : 2 Nexthops
      10.201.201.201 [HRW:P:7fff:T]
      10.250.250.250 [HRW:P:00:T][5995.5995.5992]
   Service Carving Synchronization:
      Mode           : NTP_SCT
      Peer Updates   :
           10.201.201.201 [SCT: 2021-10-17 01:25:16.1634459]
           10.250.250.250 [SCT: 2021-10-18 19:43:45.1634611]
Service Carving Results:
      Forwarders     : 1
      **Elected        : 1**
             **EVI E   :    41001**
      Not Elected    : 0
   EVPN-VPWS Service Carving Results:
      Primary        : 0
      Backup         : 0
      Non-DF         : 0
   MAC Flushing mode : STP-TCN
   Peering timer     : 3 sec [not running]
   Recovery timer    : 30 sec [not running]
   Carving timer     : 0 sec [not running]
   HRW Reset timer   : 5 sec [not running]
   Local SHG label   : 47276
   Remote SHG labels : 1
               35041 : nexthop 10.250.250.250
   Access signal mode: Bundle OOS (Default)
```

Verify the main port carving details.

```
Router# show evpn ethernet-segment interface bundle-Ether 5555 carving detail

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
N/A                      BE5555                             10.201.201.201
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether5555
     Interface MAC  : fe1d.1d8d.d489
     IfHandle       : 0x20017a84
     State          : Up
     Redundancy     : Not Defined
  VES Main port     :
     Grouping MAC   : 5995.5995.5991
     Subif count    : 30

  ESI type          : Invalid
  ES Import RT      : 0000.0000.0000 (Incomplete Configuration)
  Source MAC        : 0000.0000.0000 (PBB BSA, no ESI)
  Topology          :
     Operational    : SH
     Configured     : All-active (AApF) (default)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
  Peering Details   : 1 Nexthops
     10.201.201.201 [MOD:P:7fff]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
          10.201.201.201 [SCT: N/A]
  Service Carving Results:
     Forwarders     : 1
     **Elected        : 1**
            **EVI E   :    15001**
     Not Elected    : 0
```

Verify the evi/ead gES/EAD route and internal-label.

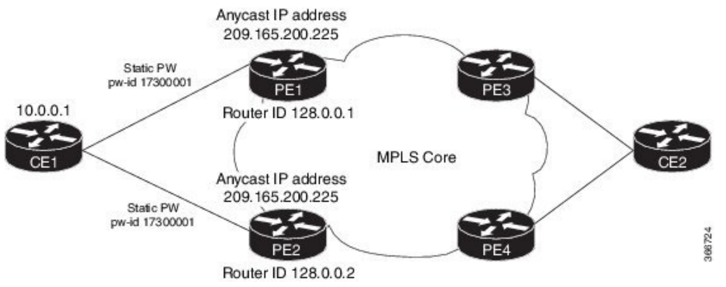
```
Router# show evpn evi vpn-id 41001 ead

VPN-ID     Encap  Ethernet Segment Id      EtherTag   Nexthop
    Label    SID
---------- ------ ----------------------- ---------- --------------------------------------
 -------- --------------------------------------
41001      MPLS   0055.5555.aabb.0000.0001 0x0        ::
           47069

10.250.250.250                          34950
41001      MPLS   0055.5555.aabb.0000.0001 0xffffffff 10.250.250.250
    0
41001      MPLS   0055.5555.ccdd.0000.0001 0x0        10.240.240.240
    56530
41001      MPLS   0055.5555.ccdd.0000.0001 0xffffffff 10.240.240.240
       0
41001      MPLS   0359.9559.9559.92ff.ffff 0xffffffff       10.250.250.250
             0
41001      MPLS   0370.e422.60e0.1eff.ffff 0xffffffff      10.240.240.240
             0


Router# show evpn internal-label vpn-id 41001 detail

VPN-ID     Encap  Ethernet Segment Id         EtherTag     Label
---------- ------ --------------------------- ----------   --------
41001      MPLS   0370.e422.60e0.1eff.ffff    0                   None
   Multi-paths resolved: FALSE (Remote all-active)
    Reason: No valid MAC paths
   Multi-paths Internal label: None
    EAD/ES      10.240.240.240                            0

41001      MPLS   0055.5555.aabb.0000.0001    0             97041
   Multi-paths resolved: TRUE (Remote all-active)
   Multi-paths Internal label: 97041
    EAD/ES      10.250.250.250                            0
                 Grouping MAC: 5995.5995.5992
    EAD/EVI     10.250.250.250                                         34950
   Summary pathlist:
 0x02000005 (P) 10.250.250.250                                    34950

41001      MPLS   0055.5555.ccdd.0000.0001    0            70349
   Multi-paths resolved: TRUE (Remote all-active)
   Multi-paths Internal label: 70349
    EAD/ES      10.240.240.240                            0
                 Grouping MAC: 70e4.2260.e01e
    EAD/EVI     10.240.240.240                                        56530
   Summary pathlist:
 0x02000004 (P) 10.240.240.240                                      56530

41001      MPLS   0359.9559.9559.92ff.ffff    0                 None
   Multi-paths resolved: FALSE (Remote all-active)
    Reason: No valid MAC paths
   Multi-paths Internal label: None
    EAD/ES      10.250.250.250                             0
```

EVPN Anycast Gateway All-Active Static Pseudowire

The EVPN Anycast Gateway All-active Static Pseudowire (PW) feature enables all-active multi-homing support for static PWs. When static PWs are configured, it overrides the default behavior of single-active, and the node becomes all-active per flow (AApF).

Configure EVPN Anycast All-active Static Pseudowire

Consider a traffic flow from CE1 to CE2. CE1 sends the traffic to PE1 or PE2. PE1 and PE2 are connected to CE1 through static PW. CE1 sends the traffic to the PEs using the same anycast IP address, and uses IGP ECMP for load balancing. Anycast PWs are static. You can configure an ESI per static PW. PE1 and PE2 forward the traffic based on the type of traffic.



Consider PE1 to be a DF and PE2 a non-DF. When a Broadcast, Unknown unicast and Multicast (BUM) traffic is sent from CE1 to PE1 or PE2. PE1 sends traffic to all other nodes towards the core side, including PE2. However, PE2 drops the traffic as it a non-DF. Similarly, PE2 sends traffic to all other nodes towards the core side, including PE1. However, PE1 drops the traffic as it is coming from a non-DF node. PE1 or PE2 sends the traffic to CE2 through MPLS core.

When BUM traffic is sent from the core side, that is from PE3 or PE4 to CE1. PE3 or PE4 sends the traffic to PE1 and PE2. PE1 forwards the traffic to CE1. PE2 drops the packets as it is a non-DF.

When unicast traffic is sent from CE1 to PE1 and PE2, both PE1 and PE2 forward the traffic to the core. When unicast traffic is sent from PE3 or PE4 to CE1, both PE1 and PE2 send the traffic to CE1.

Configure Static PW

This section describes how you can configure static PW.

```
/* Configure PE1 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-bg-bd)# neighbor 10.0.0.1 pw-id 17300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# mpls static label local 1000 remote 2000
```

```
/* Configure EVPN */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# virtual neighbor 10.0.0.1 pw-id 17300001
RP/0/RSP0/CPU0:router(config-evpn-ac-pw)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# identifier type  0 14.14.00.00.00.01.00.00.03
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# commit

/* Configure PE2 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bg1
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bd1
RP/0/RSP0/CPU0:router(config-bg-bd)# neighbor 10.0.0.1 pw-id 17300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# mpls static label local 1000 remote 2000

/* Configure EVPN */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# virtual neighbor 10.10.0.1 pw-id 17300001
RP/0/RSP0/CPU0:router(config-evpn-ac-pw)# ethernet-segment
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# identifier type  0 14.14.00.00.00.01.00.00.03
RP/0/RSP0/CPU0:router(config-evpn-ac-pw-es)# commit

/* Configure CE1 */
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group 73
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain 73-1
RP/0/RSP0/CPU0:router(config-bg-bd)# neighbor 209.165.200.225 pw-id 17300001
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# mpls static label local 2000 remote 1000
RP/0/RSP0/CPU0:router(config-bg-bd-pw)# commit
```

Running Configuration

This section shows static PW running configuration.

```
/* On PE1 */
!
configure
l2vpn
 bridge group bg1
  bridge-domain bd1
   neighbor 10.0.0.1 pw-id 17300001
    mpls static label local 1000 remote 2000

!

evpn
  virtual neighbor 10.0.0.1 pw-id 17300001
   ethernet-segment
    identifier type 0 14.14.00.00.00.01.00.00.03
    !


/* On PE2 */
!
configure
l2vpn
 bridge group bg1
  bridge-domain bd1
   neighbor 10.0.0.1 pw-id 17300001
    mpls static label local 1000 remote 2000
```

```
!

evpn
  virtual neighbor 10.0.0.1 pw-id 17300001
   ethernet-segment
    identifier type 0 14.14.00.00.00.01.00.00.03
    !

/* On CE1 */
!
configure
l2vpn
 bridge group bg73
  bridge-domain bd73-1
   neighbor 209.165.200.225 pw-id 17300001
    mpls static label local 2000 remote 1000

!
```

Verification

The outputs in this section show the number of static PWs configured on CE1, PE1, and PE2 and the configuration details of their neighbors.

```
/* CE1 static PW configuration details */

RP/0/RSP0/CPU0:router-CE1# show l2vpn bridge-domain bd-name bd-73-1
Fri Aug 11 12:36:12.732 EDT
Legend: pp = Partially Programmed.
Bridge group: bg73, bridge-domain: bd-73-1, id: 3, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 1 (1 up), VFIs: 0, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of ACs:
    BE7301.1, state: up, Static MAC addresses: 0
  List of Access PWs:
      Neighbor 128.0.0.19 pw-id 17300001, state: up, Static MAC addresses: 0
  List of VFIs:
  List of Access VFIs:

RP/0/RSP0/CPU0:router-CE1#show l2vpn bridge-domain bd-name bd-73-1 detail
Fri Aug 11 12:36:27.136 EDT
Number of groups: 2, bridge-domains: 8000, Up: 8000, Shutdown: 0, Partially-
programmed: 0
Default: 8000, pbb-edge: 0, pbb-core: 0
Number of ACs: 8000 Up: 8000, Down: 0, Partially-programmed: 0
Number of PWs: 12001 Up: 12000, Down: 1, Standby: 0, Partially-programmed: 0
Number of P2MP PWs: 0, Up: 0, Down: 0, other-state: 0
Number of VNIs: 0, Up: 0, Down: 0, Unresolved: 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 down (legacy)
    MAC withdraw relaying (access to access): disabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown unicast: enabled
  MAC aging time: 300 s, Type: inactivity
  MAC limit: 4000, Action: none, Notification: syslog
```

```
 MAC limit reached: no
MAC port down flush: enabled
 MAC Secure: disabled, Logging: disabled
 Split Horizon Group: none
 Dynamic ARP Inspection: disabled, Logging: disabled
 IP Source Guard: disabled, Logging: disabled
 DHCPv4 Snooping: disabled
 DHCPv4 Snooping profile: none
 IGMP Snooping: disabled
 IGMP Snooping profile: none
 MLD Snooping profile: none
 Storm Control: disabled
 Bridge MTU: 1500
 MIB cvplsConfigIndex: 4
 Filter MAC addresses:
 P2MP PW: disabled
 Create time: 08/08/2017 17:19:31 (2d19h 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: Bundle-Ether7301.1, state is up
     Type VLAN; Num Ranges: 1
     Rewrite Tags: []
     VLAN ranges: [1, 1]
     MTU 8986; XC ID 0xc0003e82; interworking none
     MAC learning: enabled
     Flooding:
       Broadcast & Multicast: enabled
       Unknown unicast: enabled
     MAC aging time: 300 s, Type: inactivity
     MAC limit: 4000, Action: none, Notification: syslog
     MAC limit reached: no
     MAC port down flush: enabled
     MAC Secure: disabled, Logging: disabled
     Split Horizon Group: none
     Dynamic ARP Inspection: disabled, Logging: disabled
     IP Source Guard: disabled, Logging: disabled
     DHCPv4 Snooping: disabled
     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
 List of Access PWs:
   PW: neighbor 128.0.0.19, PW ID 17300001, state is up
     PW class not set, XC ID 0xa0000013
     Encapsulation MPLS, protocol none
     Source address 10.0.0.1
     PW type Ethernet, control word disabled, interworking none
     PW backup disable delay 0 sec
     Sequencing not set
```

```
        MPLS         Local                          Remote
        ------------ ------------------------------ ---------------------------
        Label        2000                           1000
        Interface    Access PW
        MTU          1500
        Control word disabled
        PW type      Ethernet
        VCCV CV type 0x2
                     (LSP ping verification)
        VCCV CC type 0x6
                     (router alert label)
                     (TTL expiry)
        ------------ ------------------------------ ---------------------------
      MIB cpwVcIndex: 2684354579
      Create time: 08/08/2017 17:19:33 (2d19h ago)
      Last time status changed: 11/08/2017 11:39:50 (00:56:46 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: 4000, Action: none, Notification: syslog
      MAC limit reached: no
      MAC port down flush: enabled
      MAC Secure: disabled, Logging: disabled
      Split Horizon Group: none
      DHCPv4 Snooping: disabled
      DHCPv4 Snooping profile: none
      IGMP Snooping: disabled
      IGMP Snooping profile: none
      MLD Snooping profile: none
      Storm Control: bridge-domain policer
  List of VFIs:
  List of Access VFIs:

/* PE1 static PW configuration details */

RP/0/RSP0/CPU0:router-PE1#show evpn ethernet-segment esi 0 14.14.00.00.00.01.00.00.03 carving
 detail
Fri Aug 11 12:47:30.981 EDT
Legend:
  A   - Load-balancing mode and Access Protection incompatible,
  B   - No Forwarders EVPN-enabled,
  C   - Backbone Source MAC missing (PBB-EVPN),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0014.1400.0000.0100.0003 PW:10.0.0.1,17300001               128.0.0.1
                                                            128.0.0.2
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Virtual Access    :
     Name           : PW_10.0.0.1_17300001
     State          : Up
     Num PW Up      : 1
  ESI type          : 0
     Value          : 14.1400.0000.0100.0003
  ES Import RT      : 1414.0001.0003 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : All-active (AApF)
  Primary Services  : Auto-selection
  Secondary Services: Auto-selection
  Service Carving Results:
     Forwarders     : 1
     Permanent      : 0
     Elected        : 1
            EVI E   :        1
     Not Elected    : 0
  MAC Flushing mode : Invalid
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 32096
  Remote SHG labels : 1
              32096 : nexthop  128.0.0.1

/* PE2 static PW configuration details */

RP/0/RSP0/CPU0:router-PE2#show evpn ethernet-segment esi 0014.1400.0000.0100.0003 carving
detail
Legend:
  A   - Load-balancing mode and Access Protection incompatible,
  B   - No Forwarders EVPN-enabled,
  C   - Backbone Source MAC missing (PBB-EVPN),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0014.1400.0000.0100.0003 PW:10.0.0.1,17300001               128.0.0.2
                                                            128.0.0.1
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
```

```
Virtual Access    :
   Name           : PW_10.0.0.1_17300001
   State          : Up
   Num PW Up      : 1
ESI type          : 0
   Value          : 14.1400.0000.0100.0003
ES Import RT      : 1414.0001.0003 (from ESI)
Source MAC        : 0000.0000.0000 (N/A)
Topology          :
   Operational    : MH
   Configured     : All-active (AApF)
Primary Services  : Auto-selection
Secondary Services: Auto-selection
Service Carving Results:
   Forwarders     : 1
   Permanent      : 0
   Elected        : 0
   Not Elected    : 1
          EVI NE  :        1
MAC Flushing mode : Invalid
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Local SHG label   : 32096
Remote SHG labels : 1
            32096 : nexthop 128.0.0.2
```

CFM Support for EVPN

Ethernet Connectivity Fault Management (CFM) is a service-level OAM protocol that provides tools for monitoring and troubleshooting end-to-end Ethernet services per VLAN. This includes proactive connectivity monitoring, fault verification, and fault isolation. CFM can be deployed in an EVPN network. You can monitor the connections between the nodes using CFM in an EVPN network.

Restrictions

CFM for EVPN is supported with the following restrictions:

- In an active-active multi-homing scenario, when monitoring the connectivity between a multi-homed CE device and the PE devices to which it is connected, CFM can only be used across each individual link between a CE and a PE. Attempts to use CFM on the bundle between CE and PE devices cause sequence number errors and statistical inaccuracies.
- There is a possibility of artefacts in loopback and linktrace results. Either a loopback or linktrace may report multiple results for the same instance, or consecutive instances of a loopback and linktrace between the same two endpoints may produce different results.

For more information about Ethernet Connectivity Fault Management (CFM), refer to the *Configuring Ethernet OAM* chapter in the *Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Configuration Guide*.

EVPN Multiple Services per Ethernet Segment

EVPN Multiple Services per Ethernet Segment feature allows you to configure multiple services over single Ethernet Segment (ES). Instead of configuring multiple services over multiple ES, you can configure multiple services over a single ES.

You can configure the following services on a single Ethernet Bundle; you can configure one service on each sub-interface.

- EVPN-VPWS Xconnect service. Only all-active multihoming is supported.

  For more information, see *EVPN Virtual Private Wire Service (VPWS)* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*.

- Native EVPN with Integrated Routing and Bridging (IRB) on a single ES. Both single-active and all-active multihoming modes are supported. However, both single-active and all-active multihoming cannot be configured on a single ES. You can configure either single-active or all-active multihoming mode on a single ES. But, they can coexist.

  For more information, see *Configure EVPN IRB* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*.

- Native EVPN. Both single-active and all-active multihoming modes are supported. However, both single-active and all-active multihoming cannot be configured on a single ES. You can configure either single-active or all-active multihoming mode on a single ES. But, they can coexist.

  For more information see, *EVPN Features* chapter in *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*.

Configure EVPN Multiple Services per Ethernet Segment

Consider a customer edge (CE) device connected to two provider edge (PE) devices through Ethernet Bundle interface 22001. Configure multiple services on Bundle Ethernet sub-interfaces.

Configuration Example

Consider Bundle-Ether22001 ES, and configure multiple services on sub-interface.

```
/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */

Router# configure
Router(config)# interface Bundle-Ether22001.11 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q 11
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

Router# configure
Router(config)# interface Bundle-Ether22001.21 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 1 second-dot1q  21
Router(config-l2vpn-subif)# rewrite ingress tag pop 2 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

Router# configure
```

```
Route(config)# l2vpn
Router(config-l2vpn)# xconnect group xg22001
Router(config-l2vpn-xc)# p2p evpn-vpws-mclag-22001
Router(config-l2vpn-xc-p2p)# interface Bundle-Ether22001.11
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 22101 target 220101 source 220301
Router(config-l2vpn-xc-p2p)# commit
Router(config-l2vpn-xc-p2p)# exit

Router # configure
Router (config)# l2vpn
Router (config-l2vpn)# bridge group native_evpn1
Router (config-l2vpn-bg)#  bridge-domain bd21
Router (config-l2vpn-bg-bd)# interface Bundle-Ether22001.21
Router (config-l2vpn-bg-bd-ac)# routed interface BVI21
Router (config-l2vpn-bg-bd-bvi)# evi 22021
Router (config-l2vpn-bg-bd-bvi)# commit
Router (config-l2vpn-bg-bd-bvi)# exit

/* Configure Native EVPN */

Router # configure
Router (config)# evpn
Router (config-evpn)# interface Bundle-Ether22001
Router (config-evpn-ac)# ethernet-segment identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
Router (config-evpn-ac-es)# bgp route-target 2200.0001.0001
Router (config-evpn-ac-es)# exit
Router (config-evpn)# evi 24001
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:24001
Router (config-evpn-evi-bgp)# route-target export 64:24001
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 21006
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target route-target 64:10000
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22101
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64:22101
Router (config-evpn-evi-bgp)# route-target export 64:22101
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22021
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64: 22021
Router (config-evpn-evi-bgp)# route-target export 64: 22021
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# exit
Router (config-evpn-evi)# advertise-mac
Router (config-evpn-evi)# exit
Router (config-evpn)# evi 22022
Router (config-evpn-evi)# bgp
Router (config-evpn-evi-bgp)# route-target import 64: 22022
Router (config-evpn-evi-bgp)# route-target export 64: 22022
Router (config-evpn-evi-bgp)# exit
Router (config-evpn-evi)# advertise-mac
Router (config-evpn-evi)# commit
Router (config-evpn-evi)# exit
```

Running Configuration

```
/* Configure EVPN-VPWS xconnect service and native EVPN with IRB */

interface Bundle-Ether22001.11 l2transport
  encapsulation dot1q 1 second-dot1q 11
  rewrite ingress tag pop 2 symmetric
 !
interface Bundle-Ether22001.21 l2transport
  encapsulation dot1q 1 second-dot1q  21
  rewrite ingress tag pop 2 symmetric
 !
!
l2vpn
xconnect group xg22001
p2p evpn-vpws-mclag-22001
 interface Bundle-Ether22001.11
 neighbor evpn evi 22101 target 220101 source 220301
!
bridge group native_evpn1
   bridge-domain bd21
   interface Bundle-Ether22001.21
    routed interface BVI21
     evi 22021
  !
/* Configure Native EVPN */
Evpn
 interface Bundle-Ether22001
  ethernet-segment   identifier type 0 ff.ff.ff.ff.ff.ff.ff.ff.ee
  bgp route-target 2200.0001.0001
  !
  evi 24001
   bgp
    route-target import 64:24001
    route-target export 64:24001
   !
   evi 21006
    bgp
      route-target 64:100006
   !
    evi 22101
     bgp
       route-target import 64:22101
       route-target export 64:22101
      !
   evi 22021
    bgp
      route-target import 64:22021
      route-target export 64:22021
    !
    advertise-mac
  !
  evi 22022
   bgp
    route-target import 64:22022
    route-target export 64:22022
   !
    advertise-mac
  !
```

Verification

Verify if each of the services is configured on the sub-interface.

```
Router# show l2vpn xconnect summary
Number of groups: 6
Number of xconnects: 505  Up: 505  Down: 0  Unresolved: 0 Partially-programmed: 0
AC-PW: 505  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


Router# show l2vpn xconnect-service summary
Number of flexible xconnect services: 74
 Up: 74

Router# show l2vpn xconnect group xg22001 xc-name evpn-vpws-mclag-22001
Fri Sep 1 17:28:58.259 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
------------------------ ---------------------------- ------------------------------------
xg22001  evpn-vpws-mclag-22001   UP    BE22001.101  UP      EVPN 22101, 220101,64.1.1.6 UP
------------------------------------------------------------------------------------------
```

Associated Commands

- evpn
- evi
- ethernet-segment
- advertise-mac
- show evpn ethernet-segment
- show evpn evi
- show evpn summary
- show l2vpn xconnect summary
- show l2vpn xconnect group

EVPN VXLAN Ingress Replication

The EVPN VXLAN Ingress Replication feature enables the VXLAN tunnel endpoint (VTEP) to exchange local and remote VTEP IP addresses on the Virtual Network Identifier (VNI) in order to create the ingress replication list. This enables VTEPs to send and receive broadcast, unknown unicast and multicast (BUM) traffic for the VNI. These IP addresses are exchanged between VTEPs through the BGP EVPN control plane using EVPN Route Type 3. This feature enables in reduced traffic flooding, increased load sharing at VTEP, faster convergence during link and device failures, and simplified data center automation.

The VXLAN imposition node maintains a list of remote VTEP nodes that serve the same tenant VNI. Each copy of VXLAN packet is sent to the destination VTEP through underlay L3 unicast transport. EVPN Route Type 3 which is a inclusive multicast route, is used to build a replication list of VXLAN data plane VTEPs. The imposition node replicates BUM traffic for each remote VTEP node discovered by this route. Each copy of VXLAN is sent to destination VTEP through underlay L3 unicast transport. The ASR 9000 router is a DC edge router, which works as DCI gateway by stitching two MP-BGP control planes, one on the DC side, and the other on the MPLS WAN side.

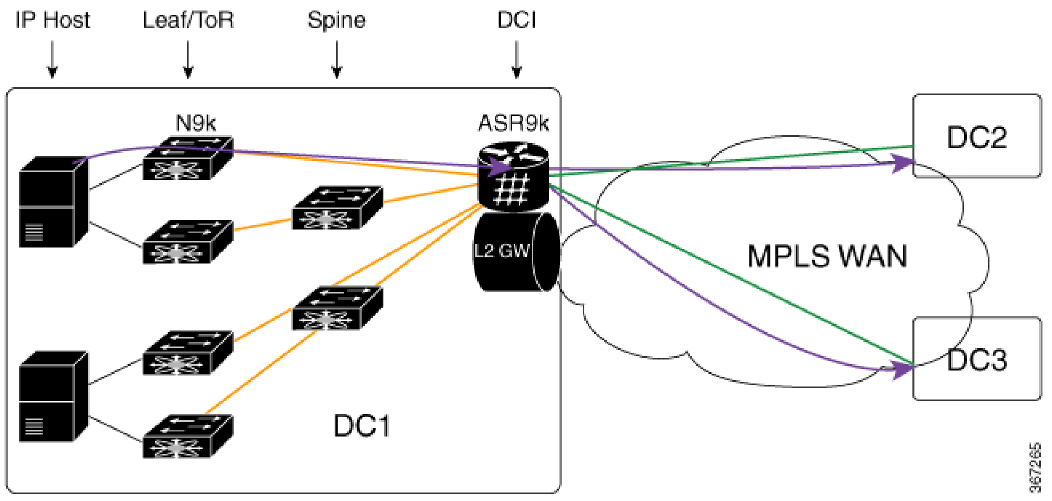
Following are the use cases of this feature:

- Single Homing VXLAN L2 gateway
- Anycast VXLAN L2 gateway
- All-active multihoming VXLAN L2 gateway

Single Homing VXLAN L2 GW

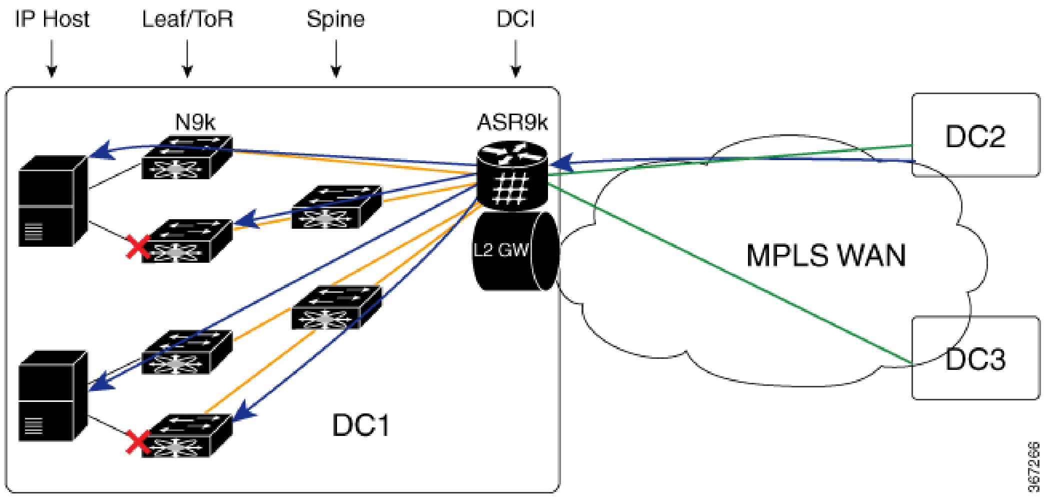
Consider a topology of single homing L2 gateway between DC and WAN. In this topology, ASR 9000 router is the DCI PE router. The L2 gateway on the PE is a bridge which forwards L2 frames between VXLAN DC and MPLS WAN. DC fabric devices, such as leaf and spine nodes, do not run IP multicast protocols, such as PIM-SM. All L2 BUM traffic between Nexus 9000 router and ASR 9000 router is forwarded through ingress replication at VXLAN imposition node.

Figure 90: Single Homing VXLAN L2 GW



A tenant VNI is enabled on all the four Nexus 9000 leaf nodes and one ASR 9000 border leaf node for L2VPN service. An IP host in DC1 initiates a communication to another IP host in DC2. The first ARP request goes from DC1 to DC2. Nexus 9000 router receives the ARP first, and uses ingress replication approach to flood the frame to other leaf nodes in DC1. One copy arrives on border leaf node ASR 9000. ASR 9000 performs L2 gateway operation. It replicates traffic using per EVI replication list at MPLS WAN side. One copy is sent to DC2. The other to DC3.

In the reverse direction, when an IP host in DC2 initiates a communication with an IP host in DC1, an ARP request arrives at ASR 9000 DCI PE from WAN. ASR 9000 performs L2 gateway operation using per VNI ingress replication list for VXLAN. A total of four copies are created. Each copy is sent to one Nexus 9000 leaf node. Nexus 9000 leaf nodes that are configured as DFs forward the traffic to IP hosts on VMs.



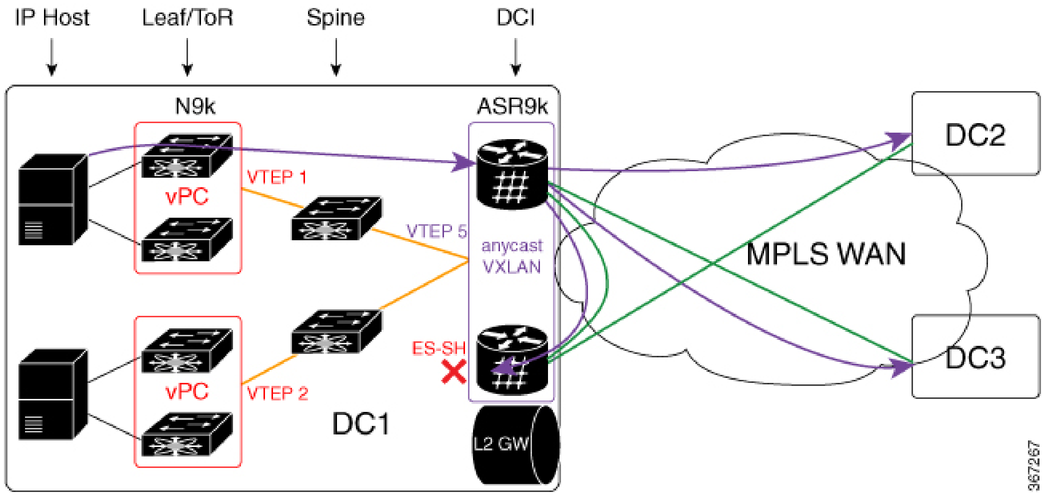
Anycast VXLAN L2 Gateway

Anycast VXLAN L2 gateway requires multihoming gateway nodes to use a common VTEP IP address. Gateway nodes in the same DC advertise the common VTEP IP in all EVPN routes from type 2 to type 5. Nexus 9000 leaf nodes in the DC considers only one border leaf VTEP located on multiple physical gateway nodes. Each Nexus 9000 router forwards traffic to the nearest gateway node through IGP routing.

Among multihoming DCI gateway nodes, an EVPN Ethernet segment is created on VXLAN facing NVE interface. One of the nodes is elected as DF for a tenant VNI. The DF node floods BUM traffic from WAN to DC. All DCI PE nodes discover each other through EVPN inclusive multicast routes advertised through WAN.

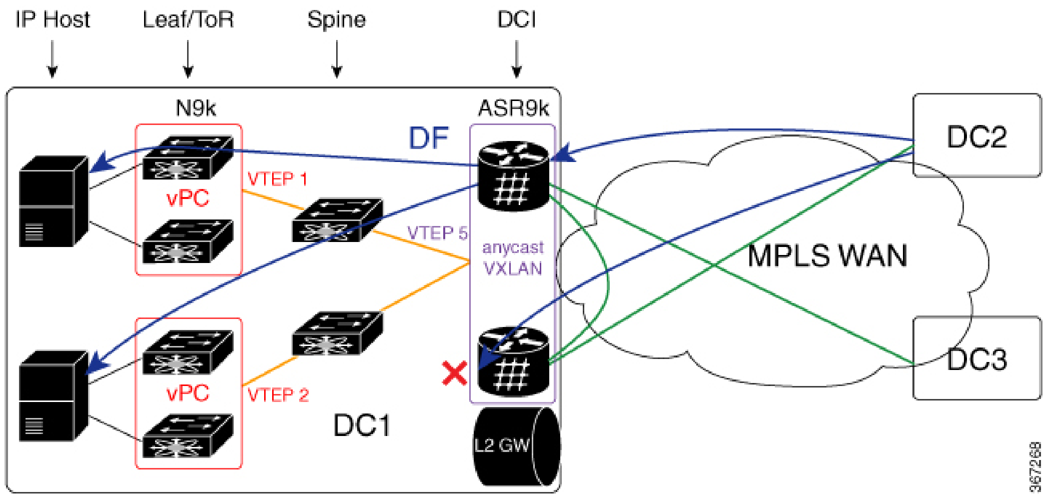
Consider a topology of anycast VXLAN L2 gateway between DC and WAN. In this topology, both ASR 9000 PE nodes share the same source VTEP IP address (VTEP5). Nexus 9000 router runs in vPC mode. ASR 9000 nodes advertise inclusive multicast routes using VTEP5 IP address. Nexus 9000 leaf nodes discover only one VTEP hosted by two ASR 9000 nodes.

Figure 91: Anycast VXLAN L2 Gateway



When the Nexus 9000 router in DC1 receives BUM traffic from local IP host, it sends one copy to VTEP5. IGP routing in underlay transport chooses the nearest ASR 9000 router as the destination. After ASR 9000 router receives the L2 frame, it replicates it to MPLS WAN side. Three copies are sent to WAN. One arrives on peer ASR 9000 router in the same DC. The copy is dropped on peer PE using Ethernet Segment Split-Horizon feature.

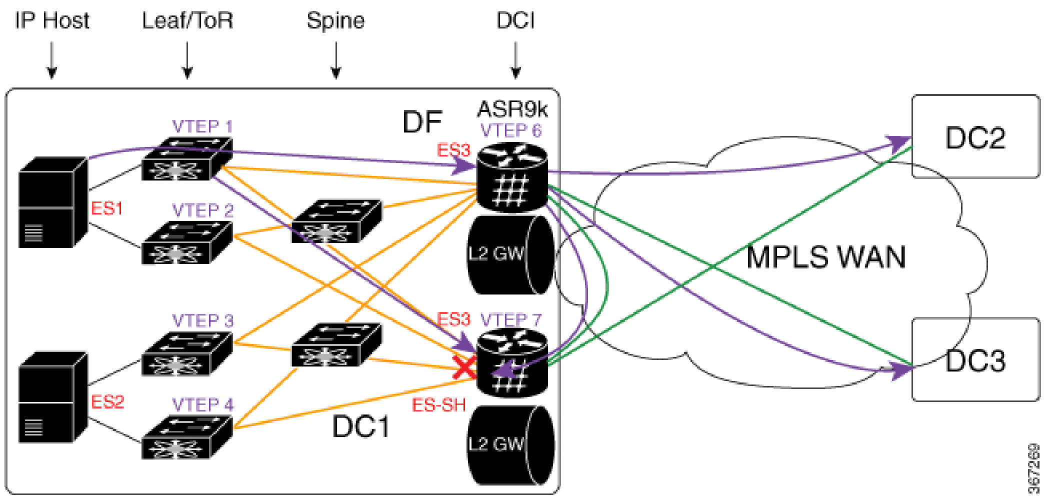
In the direction from DC2 and DC3 to DC1, both ASR 9000 DCI PE nodes receive the same BUM traffic from MPLS WAN. The DF PE for the tenant VNI forwards traffic to DC1. Non-DF PE drops BUM traffic from WAN.



All-Active Multihoming VXLAN L2 Gateway

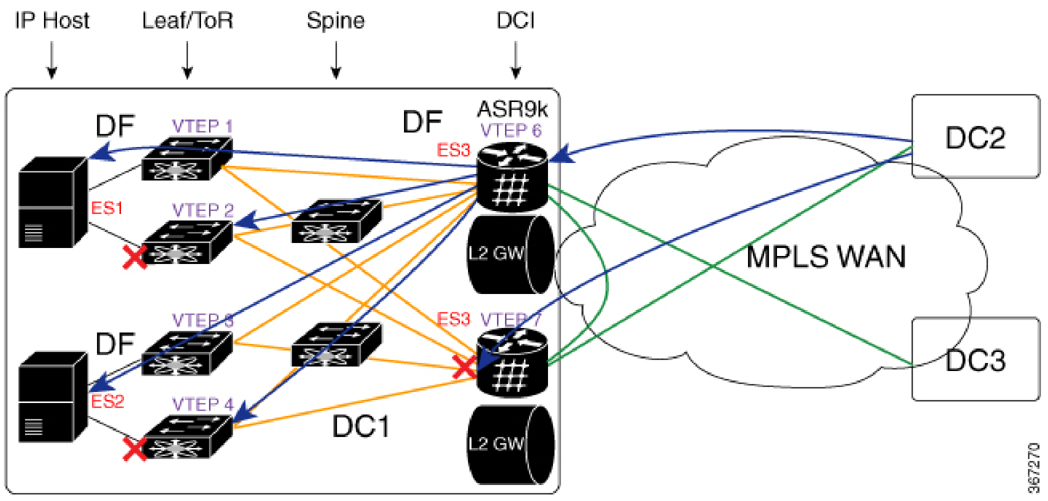
Consider a topology of all-active multihoming VXLAN L2 gateway where all leaf nodes, including Nexus 9000 node and ASR 9000 node, each has an unique VTEP IP address. Each Nexus 9000 leaf node creates EVPN Ethernet segment (ES1 and ES2) for dual-homed VM server. ASR 9000 border leaf nodes create an Ethernet Segment (ES3) for VXLAN facing NVE interface. Since every leaf node advertises inclusive multicast route using its local VTEP IP, ASR 9000 node receives four routes from Nexus 9000 node. The per VNI ingress replication list includes four remote VTEP (VTEP1 to VTEP4). Every Nexus 9000 node receives two routes from ASR 9000 gateway nodes. It sends BUM traffic to both ASR 9000 nodes. To prevent traffic duplication, only one of the ASR 9000 nodes can accept VXLAN traffic from Nexus 9000 leaf using DF. DF election is done at per tenant VNI level. One half of the VNIs elect top PE as DF. The other half elect bottom PE. DF PE accepts traffic both from DC and WAN. Non-DF drops traffic from DC and WAN.

Figure 92: All-Active Multihoming VXLAN L2 Gateway



BUM traffic from DC1 arrives at Nexus 9000 leaf first. Nexus 9000 replicates the traffic to two ASR 9000 DCI nodes. DF DCI nodes flood traffic to WAN. Non-DF node drops traffic from DC fabric. Traffic flooded to WAN goes to DC2 and DC3. One copy comes back to DC1 through bottom DCI node. The bottom DCI node compares the split horizon label in the received MPLS packet and drops the packet.

In the reverse direction, when the traffic flows from DC2 and DC3, towards DC1, arrives at both top and bottom DCI nodes. The bottom DCI which is a non-DF drops traffic. The top DCI which is a DF, forwards four copies to remote leaf nodes. The Nexus 9000 leaf nodes forward traffic to an IP host.



Configure EVPN VXLAN Ingress Replication

Perform the following tasks to configure EVPN VXLAN Ingress Replication feature:

- Configure DCI
- Configure ToR

```
/* DCI Configuration */

/* Configure Network Virtualization Endpoint (NVE) Interface */

Router# configure
Router(config)# interface nve 40
Router(config-if)# member vni 40002
Router(config-if)# host-reachability protocol bgp
Router(config-if)# source-interface loopback 40
Router(config-if)# anycast source-interface Loopback41
Router(config-if)# ingress-replication protocol bgp
Router(config-if)# commit

/* Configure a Bridge Domain */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd2
Router(config-l2vpn-bg-bd)# evi 40
Router(config-l2vpn-bg-bd-evi)# exit
Router(config-l2vpn-bg-bd)# member vni 40002
Router(config-l2vpn-bg-bd-vni)# commit

/* Configure Ethernet Segment Identifier */

Router# configure
Router(config)# evpn
Router(config-evpn)# interface nve 40
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 28.28.28.00.00.40.00.00.13
Router(config-evpn-ac-es)# bgp route-target 200:40000 stitching
Router(config-evpn-ac-es)# commit

/* Configure the routing sessions between the DCI and ToR */

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 192.168.0.4
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 15.15.15.5 -------> ToR ebgp neighbour
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# ebgp-multihop 255
Router(config-bgp-nbr)# address-fmaily l2vpn evpn
Router(config-bgp-nbr-af)# import stitching-rt reoriginate
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# encapsulation-type vxlan
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# advertise l2vpn evpn re-originated stitching-rt
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp)# neighbor 192.168.0.2 -------> DCI BGP neighbour
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-fmaily l2vpn evpn
Router(config-bgp-nbr-af)# import stitching-rt reoriginate
Router(config-bgp-nbr-af)# advertise l2vpn evpn re-originated stitching-rt
Router(config-bgp-nbr-af)# commit
```

```
/* ToR Configuration */

/* Configure Network Virtualization Endpoint (NVE) Interface */

Router# configure
Router(config)# interface nve 40
Router(config-if)# member vni 40002
Router(config-if)# host-reachability protocol bgp
Router(config-if)# source-interface loopback 40
Router(config-if)# anycast source-interface Loopback41
Router(config-if)# ingress-replication protocol bgp
Router(config-if)# commit

/* Configure RD and Route Targets for VXLAN Bridging */

Router# configure
Router(config)# evpn
Router(config-evpn)# router bgp
Router(config-evpn-bgp)# rd auto
Router(config-evpn-bgp)# route-target import auto
Router(config-evpn-bgp)# route-target import 200:40000
Router(config-evpn-bgp)# route-target export 200:40000
Router(config-evpn-bgp)# commit


/* Configure the routing sessions between the ToR and DCI */

Router# configure
Router(config)# router bgp 200
Router(config-bgp)# bgp router-id  10.5.41.41
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# maximum-paths 8
Router(config-bgp-af)# maximum-paths ibgp 8
Router(config-bgp-af)# exit
!
Router(config-bgp)# 192.168.0.4 -------> DCI neighbour: ebgp
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# ebgp-multihop 255
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# address-fmaily l2vpn evpn
Router(config-bgp-nbr-af)# send-community extended
Router(config-bgp-nbr-af)# route-map passall in
Router(config-bgp-nbr-af)# route-map IR-test out
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp)# neighbor 192.168.0.2 -------> VXLAN neighbour
Router(config-bgp-nbr)# remote-as 200
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-fmaily l2vpn evpn
Router(config-bgp-nbr-af)# send-community extended
Router(config-bgp-nbr-af)# commit
```

Running Configuration

```
/* DCI Configuration */

interface nve40
 member vni 40002
  host-reachability protocol bgp
 !
 source-interface Loopback40
 anycast source-interface Loopback41
 ingress-replication protocol bgp

l2vpn
 bridge group bg1
  bridge-domain bd2
   evi 40
   member vni 40002

evpn
 interface nve 40
  ethernet-segment
   identifier type 0 28.28.28.00.00.40.00.00.13
    bgp route-target 200:40000 stitching

evpn evi 40
  bgp route-target 200:40000 stitching
router bgp 100
 bgp router-id 192.168.0.4
 address-family l2vpn evpn
 !
 neighbor 15.15.15.5 -------> TOR ebgp neighbor
  remote-as 200
  ebgp-multihop 255
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass-all in
   encapsulation-type vxlan
   route-policy pass-all out
   next-hop-self
   advertise l2vpn evpn re-originated stitching-rt

 neighbor 192.168.0.2  ---------> DCI BGP neighor
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
   import re-originate stitching-rt
   advertise l2vpn evpn re-originated

/* ToR Configuration */

interface nve 40
 member vni 40002
 host-reachability protocol bgp
 source-interface loopback 40
 anycast source-interface Loopback41
 ingress-replication protocol bgp

evpn
 router bgp
  rd auto
  route-target import auto
  route-target import 200:40000
  route-target export 200:40000

router bgp 200
 bgp router-id  10.5.41.41
 address-family l2vpn evpn
  maximum-paths 8
  maximum-paths ibgp 8

  neighbor 192.168.0.4  -----------> DCI neighbour: ebgp
```

```
   remote-as 100
    update-source loopback0
    ebgp-multihop 255
    address-family ipv4 unicast
    address-family l2vpn evpn
      send-community extended
      route-map passall in
      route-map IR-test out

  neighbor 192.168.0.6 ------->  VXLAN neighbour
   remote-as 200
    update-source loopback0
    address-family l2vpn evpn
      send-community both
```

Verification

Verify that you have configured EVPN VXLAN Ingress Replication feature successfully.

```
DC3# show evpn evi vpn-id 40 inclusive-multicast detail
Ethernet Tag: 0, Originating IP: 192.168.0.2, vpn-id: 40
    Nexthop: 192.168.0.2
    Label  : 24004
    Source : Remote
    Encap  : MPLS
Ethernet Tag: 0, Originating IP: 192.168.0.3, vpn-id: 40
    Nexthop: 192.168.0.3
    Label  : 24003
    Source : Remote
    Encap  : MPLS
Ethernet Tag: 0, Originating IP: 192.168.0.4, vpn-id: 40
    Nexthop: ::
    Label  : 24001
    Source : Local
    Encap  : MPLS
```

```
DC2# show evpn ethernet-segment interface  nve 40 detail

Ethernet Segment Id      Interface                          Nexthops
 ------------------------ ---------------------------------- --------------------
0028.2828.0000.4000.0013 nv40                               128.0.0.1
                                                                                 128.0.0.2

ES to BGP Gates   : Ready
ES to L2FIB Gates : Ready
Main port         :
       Interface name : nve40
       Interface MAC  : 0000.0000.0000
       IfHandle       : 0x0003e960
       State          : Up
       Redundancy     : Not Defined
ESI type          : 0
       Value          : 28.2828.0000.4000.0013
ES Import RT      : 2828.2800.0040 (from ESI)
Source MAC        : 0000.0000.0000 (N/A)
Topology          :
       Operational    : MH
       Configured     : All-active (AApF) (default)
Primary Services  : Auto-selection
Secondary Services: Auto-selection
Service Carving Results:
        Forwarders     : 4000
        Permanent      : 0
        Elected        : 2000
        Not Elected    : 2000
MAC Flushing mode : Invalid
Peering timer     : 30 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Local SHG label   : 38029
Remote SHG labels : 1
        46029 : nexthop 128.0.0.1
```

```
DCI# show l2vpn forwarding protection main-interface nve 40 location 0/2/CPU0
Main Interface ID                Instance   State
-------------------------------- ---------- ------------
nve40                            0          FORWARDING
nve40                            1          FORWARDING
nve40                            2          PE2CEBLOCK
nve40                            3          FORWARDING
nve40                            4          PE2CEBLOCK
nve40                            5          FORWARDING
nve40                            6          PE2CEBLOCK
nve40                            7          FORWARDING
nve40                            8          PE2CEBLOCK
nve40                            9          FORWARDING
nve40                            10         PE2CEBLOCK
nve40                            11         FORWARDING
nve40                            12         PE2CEBLOCK
nve40                            13         FORWARDING
nve40                            14         PE2CEBLOCK
```

```
DC3# show evpn evi vpn-id 40 inclusive-multicast detail

Ethernet Tag: 0, Originating IP: 10.4.41.41, vpn-id: 40
    Nexthop: ::
    Label  : 40000
    Source : Local
    Encap  : VXLAN
Ethernet Tag: 0, Originating IP: 10.5.41.41, vpn-id: 40
    Nexthop: 10.5.41.41
    Label  : 40000
    Source : Remote
    Encap  : VXLAN
Ethernet Tag: 0, Originating IP: 10.6.41.41, vpn-id: 40
    Nexthop: 10.6.41.41
    Label  : 40000
    Source : Remote
    Encap  : VXLAN
```

```
DC3# show l2vpn forwarding bridge-domain evpn inclusive-multicast location 0/0/CPU0

Bridge-Domain Name               BD-ID  XCID       Next Hop        Label/VNI
-------------------------------- ------ ---------- --------------- ----------
l2cp-ir:l2cp-40                  1      0xfff01002 192.168.0.2     24004   ;; MPLS-side
                                                   192.168.0.3     24003
l2cp-ir:l2cp-40                  1      0xfffc1805 10.5.41.41      40000   ;; VXLAN side
                                                   10.6.41.41      40000
------------------------------------------------------------------------------------------
```

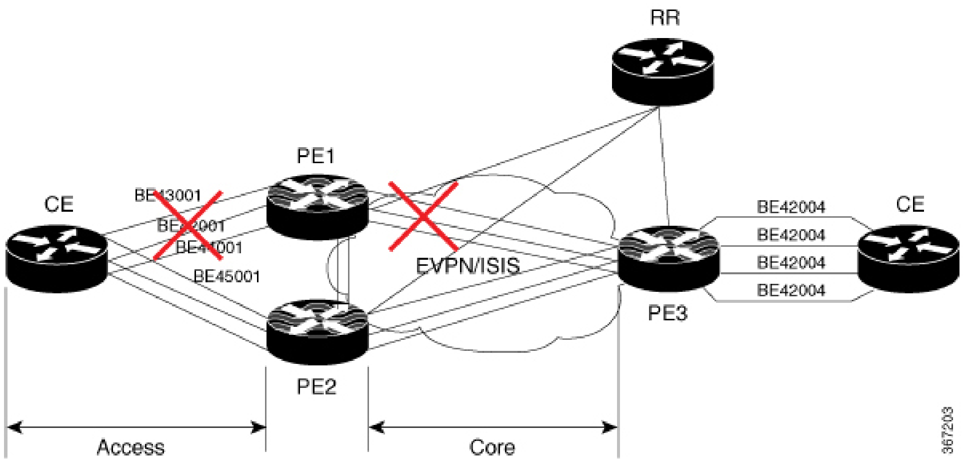
EVPN Core Isolation Protection

The EVPN Core Isolation Protection feature enables you to monitor and detect the link failure in the core. When a core link failure is detected in the provider edge (PE) device, EVPN brings down the PE's Ethernet Segment (ES), which is associated with access interface attached to the customer edge (CE) device.

EVPN replaces ICCP in detecting the core isolation. This new feature eliminates the use of ICCP in the EVPN environment.

Consider a topology where CE is connected to PE1 and PE2. PE1, PE2, and PE3 are running EVPN over the MPLS core network. The core interfaces can be Gigabit Ethernet or bundle interface.

Figure 93: EVPN Core Isolation Protection



When the core links of PE1 go down, the EVPN detects the link failure and isolates PE1 node from the core network by bringing down the access network. This prevents CE from sending any traffic to PE1. Since BGP session also goes down, the BGP invalidates all the routes that were advertised by the failed PE. This causes the remote PE2 and PE3 to update their next-hop path-list and the MAC routes in the L2FIB. PE2 becomes the forwarder for all the traffic, thus isolating PE1 from the core network.

When all the core interfaces and BGP sessions come up, PE1 advertises Ethernet A-D Ethernet Segment (ES-EAD) routes again, triggers the service carving and becomes part of the core network.

Configure EVPN Core Isolation Protection

Configure core interfaces under EVPN group and associate that group to the Ethernet Segment which is an attachment circuit (AC) attached to the CE. When all the core interfaces go down, EVPN brings down the associated access interfaces which prevents the CE device from using those links within their bundles. All interfaces that are part of a group go down, EVPN brings down the bundle and withdraws the ES-EAD route.

Restrictions

- A maximum of 24 groups can be created under the EVPN.
- A maximum of 12 core interfaces can be added under the group.
- The core interfaces can be reused among the groups. The core interface can be a bundle interface.

- EVPN group must only contain core interfaces, do not add access interfaces under the EVPN group.
- The access interface can only be a bundle interface.
- EVPN core facing interfaces must be physical or bundle main interfaces only. Sub-interfaces are not supported.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# group 42001
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/1
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/3
Router(config-evpn-group)#exit
!
Router(config-evpn)# group 43001
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/2
Router(config-evpn-group)# core interface GigabitEthernet0/2/0/4
Router(config-evpn-group)#exit
!
Router# configure
Router(config)# evpn
Router(config-evpn)# interface bundle-Ether 42001
Router(config-evpn-ac)# core-isolation-group 42001
Router(config-evpn-ac)# exit
!
Router(config-evpn)# interface bundle-Ether 43001
Router(config-evpn-ac)# core-isolation-group 43001
Router(config-evpn-ac)# commit
```

Running Configuration

```
configure
 evpn
  group 42001
   core interface GigabitEthernet0/2/0/1
   core interface GigabitEthernet0/2/0/3
   !
  group 43001
   core interface GigabitEthernet0/2/0/2
   core interface GigabitEthernet0/2/0/4
   !
 !
configure
 evpn
  interface bundle-Ether 42001
   core-isolation-group 42001
   !
  interface bundle-Ether 43001
   core-isolation-group 43001
   !
  !
```

Verification

The **`show evpn group`** command displays the complete list of evpn groups, their associated core interfaces and access interfaces. The status, up or down, of each interface is displayed. For the access interface to be up, at least one of the core interfaces must be up.

```
Router# show evpn group /* Lists specific group with core-interfaces and access interface
status */
EVPN Group: 42001
  State: Ready
  Core Interfaces:
     Bundle-Ethernet110: down
     Bundle-Ethernet111: down
     GigabethEthernet0/2/0/1: up
     GigabethEthernet0/2/0/3: up
     GigabethEthernet0/4/0/8: up
     GigabethEthernet0/4/0/9: up
     GigabethEthernet0/4/0/10: up
  Access Interfaces:
     Bundle-Ether42001: up

EVPN Group: 43001
  State: Ready
  Core Interfaces:
     Bundle-Ethernet110: down
     GigabethEthernet0/2/0/2: up
     GigabethEthernet0/2/0/4: up
     GigabethEthernet0/4/0/9: up

  Access Interfaces:
     Bundle-Ether43001: up
```

Configurable Recovery Time for EVPN Core Isolation Group

Table 16: Feature History Table

Feature Name	Release Information	Feature Description
Configurable Recovery Time for EVPN Core Isolation Group	Release 7.6.1	You can now configure the recovery time for the EVPN core isolation group after the core interfaces recover from a network failure. This functionality is important because post-failure recovery, you can provide sufficient time for the EVPN PE nodes to relearn the MAC addresses and BGP routes received from the remote PEs. There's also time to handle delays in exchanging EVPN routes after recovery. This feature introduces the **core-de-isolation** command under the EVPN Timers configuration mode.

When the core link failure is detected on the PE device, the PE device is isolated from the network and brings down the access interfaces connected to this PE till the core interfaces recover. When the core links recover, the default recovery delay timer begins. The access interfaces become active after the default recover delay timer of 60 seconds expire. The core isolation group recovery delay timer was not user-configurable.

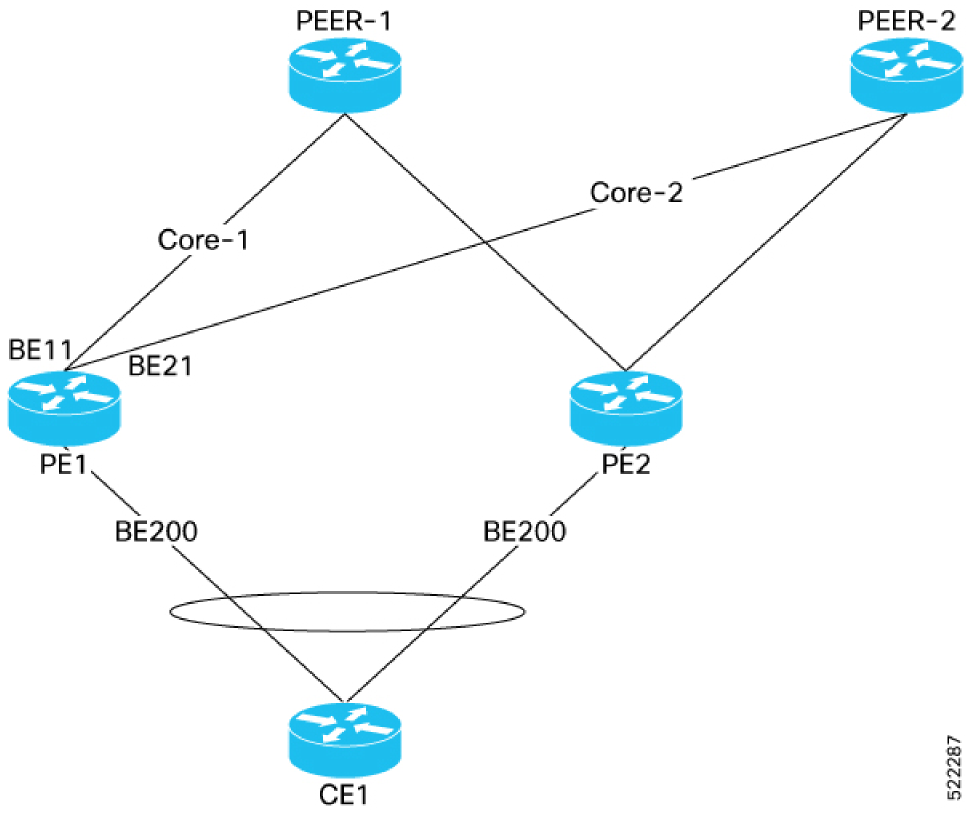
Under scale situations where a network has high MAC addresses, it is observed that the 60 seconds is too short to bring up the access bundle interface as there can be multiple reasons which can delay the exchange of EVPN routes even after the core interfaces have come up.

This feature allows you to configure the core isolation group recovery time to handle delays coming from the core and provides enough time for the EVPN PE nodes to relearn the MAC addresses. You can configure the core isolation group recovery time using the **core-de-isolation** command.

Topology

Consider a topology where CE1 is connected to PE1 and PE2. PE1 and PE2 are running EVPN over the MPLS core network. The core interfaces on PE1 are configured with BE11 and BE22. When the core links of PE1 go down, the EVPN detects the link failure and isolates the PE1 node from the core network, and brings down the access interfaces connected to PE1. This prevents CE1 from sending any traffic to PE1.

When all the core interfaces and BGP sessions come up, PE1 advertises Ethernet A-D Ethernet Segment (ES-EAD) routes again, triggers the service carving, and becomes part of the core network. The access interfaces connected to PE1 from CE1 also come up after the *core-de-isolation* timer value expires.



Configurable Recovery Time for EVPN Core Isolation Group

To enable this feature, configure core interfaces under the EVPN group and associate that group to the Ethernet Segment which is an attachment circuit (AC) attached to the CE.

Perform the following tasks to configure recovery time for EVPN core isolation group:

- Configure EVPN core interfaces on PE1
- Configure *core-de-isolation* timer on PE1
- Configure attachment circuits on CE1

Configuration Example

Configure EVPN core interfaces on PE1.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# group 100
Router(config-evpn-group)# core interface BE11
Router(config-evpn-group)# core interface BE21
Router(config-evpn-group)# commit
```

Configure core-de-isolation timer on PE1.

```
Router# configure
Router(config)# evpn timers
Router(config-evpn-timers)# core-de-isolation 120
Router(config-evpn-timers)# commit
```

Configure attachment circuits on CE1.

```
/* Configure interface Bundle-Ether200 and associate it to core isolation group 100 */
Router # configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether200
Router(config-evpn-ac)# ethernet-segment identifier type 0 11.11.11.11.11.11.11.11.11
Router(config-evpn-ac-es)# bgp route-target 1111.1111.1111
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# core-isolation-group 100

/* Configure interface Bundle-Ether201 and associate it to core isolation group 100 */
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether201
Router(config-evpn-ac)# ethernet-segment identifier type 0 11.22.22.22.22.22.22.22.22
Router(config-evpn-ac-es)# bgp route-target 1111.2222.2222
Router(config-evpn-ac-es)# exit
Router(config-evpn-ac)# core-isolation-group 100
```

Running Configuration

This section shows the EVPN core isolation group recovery delay timer running configuration.

```
/* Configure EVPN core interfaces on PE1 */
evpn
 group 100
  core interface Bundle-Ether11
  core interface Bundle-Ether21
!
!
/* Configure core-de-isolation timer on PE1 */
evpn timers
  core-de-isolation 120
!
!
/* Configure attachment circuits on CE1 */
evpn
 interface Bundle-Ether200
  ethernet-segment
   identifier type 0 11.11.11.11.11.11.11.11.11
   bgp route-target 1111.1111.1111
  !
  core-isolation-group 100
!
!
```

```
evpn
 interface Bundle-Ether201
  ethernet-segment
   identifier type 0 11.22.22.22.22.22.22.22.22
   bgp route-target 1111.2222.2222
  !
  core-isolation-group 100
!
!
```

Verification

The following output shows that all core interfaces and access interfaces are UP. The *core de-isolation* timer value is configured as 120 seconds, but not running as the core interfaces are UP.

```
Router# show evpn group
EVPN Group: 100

  state: Ready

  Core Interfaces:
     Bundle-Ether11: up
     Bundle-Ether21: up

  Access Interfaces:
     Bundle-Ether200: up
     Bundle-Ether201: up

Router# show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                     : 141
Number of TEPs                     : 2
Number of Local EAD Entries        : 178
Number of Remote EAD Entries       : 534
Number of Local MAC Routes         : 89
          MAC                      : 89
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Local ES:Global MAC      : 1
Number of Remote MAC Routes        : 0
          MAC                      : 0
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Remote SYNC MAC Routes   : 0
Number of Local IMCAST Routes      : 89
Number of Remote IMCAST Routes     : 178
Number of Internal Labels          : 178
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries               : 3
Number of Neighbor Entries         : 178
EVPN Router ID                     : 192.168.10.1
BGP ASN                            : 64600
PBB BSA MAC address                : d46a.3599.50d8
Global peering timer               :      3 seconds
Global recovery timer              :     30 seconds
Global carving timer               :      0 seconds
Global MAC postpone timer          :    300 seconds [not running]
Global core de-isolation timer     :    120 seconds [not running]
EVPN services costed out on node   : No
```

```
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

Failure Scenario

The following example shows the failure scenario and how the *core de-isolation* timer works.

Let's bring down the core interfaces:

```
Router# configure
Router(config)# interface Bundle-Ether11
Router(config-if)# shutdown
Router(config-if)# exit
Router(config)# interface Bundle-Ether21
Router(config-if)# shutdown
Router(config-if)# commit
```

This example shows when the core interfaces are shutdown even the access interfaces are down and the core is isolated.

```
Router# show evpn group

EVPN Group: 100

  state: Isolated

  Core Interfaces:
     Bundle-Ether11: shutdown
     Bundle-Ether21: shutdown

  Access Interfaces:
     Bundle-Ether200: down
     Bundle-Ether201: down
```

This example shows that the *core de-isolation timer* is not yet running because the core interfaces are still down.

```
Router# show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                     : 141
Number of TEPs                     : 0
Number of Local EAD Entries        : 178
Number of Remote EAD Entries       : 0
Number of Local MAC Routes         : 89
          MAC                      : 89
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Local ES:Global MAC      : 1
Number of Remote MAC Routes        : 0
          MAC                      : 0
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Remote SYNC MAC Routes   : 0
Number of Local IMCAST Routes      : 89
Number of Remote IMCAST Routes     : 0
Number of Internal Labels          : 0
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries               : 3
Number of Neighbor Entries         : 0
EVPN Router ID                     : 192.168.10.1
BGP ASN                            : 64600
PBB BSA MAC address                : d46a.3599.50d8
Global peering timer               :      3 seconds
Global recovery timer              :     30 seconds
Global carving timer               :      0 seconds
Global MAC postpone timer          :    300 seconds [not running]
Global core de-isolation timer     :    120 seconds [not running]
EVPN services costed out on node   : No
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

Let's bring up the core interfaces and see how the *core de-isolation* timer starts.

```
Router# rollback configuration last 1

Loading Rollback Changes.
Loaded Rollback Changes in 1 sec
Committing.
6 items committed in 1 sec (5)items/sec
Updating.
Updated Commit database in 1 sec
Configuration successfully rolled back 1 commits.
```

In this example, you can see that the *core de-isolation* timer starts running after the core interfaces come up. When the core interfaces are UP, the state of core changes to Deisolating. In the following output you can see the state as Deisolating and core interfaces are up and the *core de-isolation* timer has started.

The access interfaces come up only after the *core de-isolation* timer value expires. In the following output you can see the access interfaces are still down.

```
Router# show evpn group

EVPN Group: 100

  state: Deisolating

  Core Interfaces:
     Bundle-Ether11: up
     Bundle-Ether21: up

  Access Interfaces:
     Bundle-Ether200: down
     Bundle-Ether201: down

Router# show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                     : 141
Number of TEPs                     : 2
Number of Local EAD Entries        : 178
Number of Remote EAD Entries       : 534
Number of Local MAC Routes         : 89
          MAC                      : 89
          MAC-IPv4                 : 0
          MAC-IPv6                 : 0
Number of Local ES:Global MAC      : 1
Number of Remote MAC Routes        : 0
```

```
          MAC                       : 0
          MAC-IPv4                  : 0
          MAC-IPv6                  : 0
Number of Remote SYNC MAC Routes   : 0
Number of Local IMCAST Routes      : 89
Number of Remote IMCAST Routes     : 178
Number of Internal Labels          : 178
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries               : 3
Number of Neighbor Entries         : 178
EVPN Router ID                     : 192.168.10.1
BGP ASN                            : 64600
PBB BSA MAC address                : d46a.3599.50d8
Global peering timer               :      3 seconds
Global recovery timer              :     30 seconds
Global carving timer               :      0 seconds
Global MAC postpone timer          :    300 seconds [not running]
Global core de-isolation timer     :    120 seconds [running, 14.6 sec left]
EVPN services costed out on node   : No
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

The following output shows that the *core de-isolation* timer has expired.

```
Router# show evpn summary
-----------------------------
Global Information
-----------------------------
Number of EVIs                     : 141
Number of TEPs                     : 2
Number of Local EAD Entries        : 178
Number of Remote EAD Entries       : 534
Number of Local MAC Routes         : 89
          MAC                       : 89
          MAC-IPv4                  : 0
          MAC-IPv6                  : 0
Number of Local ES:Global MAC      : 1
Number of Remote MAC Routes        : 0
          MAC                       : 0
          MAC-IPv4                  : 0
          MAC-IPv6                  : 0
Number of Remote SYNC MAC Routes   : 0
Number of Local IMCAST Routes      : 89
Number of Remote IMCAST Routes     : 178
Number of Internal Labels          : 178
Number of single-home Internal IDs : 0
Number of multi-home Internal IDs  : 0
Number of ES Entries               : 3
Number of Neighbor Entries         : 178
EVPN Router ID                     : 192.168.10.1
BGP ASN                            : 64600
PBB BSA MAC address                : d46a.3599.50d8
Global peering timer               :      3 seconds
Global recovery timer              :     30 seconds
Global carving timer               :      0 seconds
Global MAC postpone timer          :    300 seconds [not running]
Global core de-isolation timer     :    120 seconds [not running]
EVPN services costed out on node   : No
          Startup-cost-in timer    : Not configured
          EVPN manual cost-out     : No
          EVPN Bundle Convergence  : No
```

After the *core de-isolation* timer expires, you can see that the state is Ready, and both core and access interfaces are UP.

```
Router# show evpn group

EVPN Group: 100

  state: Ready

  Core Interfaces:
     Bundle-Ether11: up
     Bundle-Ether21: up

  Access Interfaces:
     Bundle-Ether200: up
     Bundle-Ether201: up
```

EVPN Routing Policy

The EVPN Routing Policy feature provides the route policy support for address-family L2VPN EVPN. This feature adds EVPN route filtering capabilities to the routing policy language (RPL). The filtering is based on various EVPN attributes.

A routing policy instructs the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another.

This feature enables you to configure route-policies using EVPN network layer reachability information (NLRI) attributes of EVPN route type 1 to 5 in the route-policy match criteria, which provides more granular definition of route-policy. For example, you can specify a route-policy to be applied to only certain EVPN route-types or any combination of EVPN NLRI attributes. This feature provides flexibility in configuring and deploying solutions by enabling route-policy to filter on EVPN NLRI attributes.

To implement this feature, you need to understand the following concepts:

- Routing Policy Language
- Routing Policy Language Structure
- Routing Policy Language Components
- Routing Policy Language Usage
- Policy Definitions
- Parameterization
- Semantics of Policy Application
- Policy Statements
- Attach Points

For information on these concepts, see Implementing Routing Policy.

Currently, this feature is supported only on BGP neighbor "in" and "out" attach points. The route policy can be applied only on inbound or outbound on a BGP neighbor.

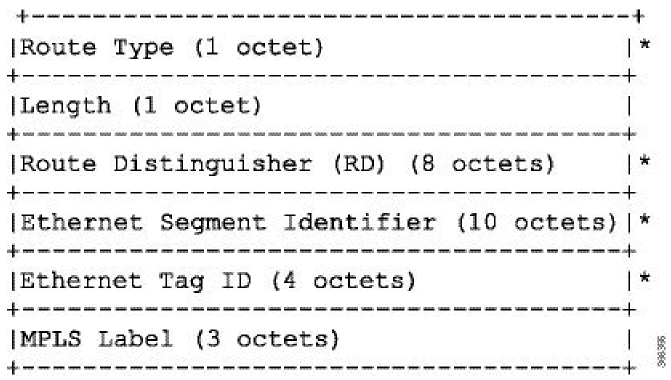
EVPN Route Types

The EVPN NLRI has the following different route types:

Route Type 1: Ethernet Auto-Discovery (AD) Route

The Ethernet (AD) routes are advertised on per EVI and per Ethernet Segment Identifier (ESI) basis. These routes are sent per Ethernet segment (ES). They carry the list of EVIs that belong to the ES. The ESI field is set to zero when a CE is single-homed.

An Ethernet A-D route type specific EVPN NLRI consists of the following fields:



```
+---------------------------------------+
|Route Type (1 octet)                   |*
+---------------------------------------+
|Length (1 octet)                       |
+---------------------------------------+
|Route Distinguisher (RD) (8 octets)    |*
+---------------------------------------+
|Ethernet Segment Identifier (10 octets)|*
+---------------------------------------+
|Ethernet Tag ID (4 octets)             |*
+---------------------------------------+
|MPLS Label (3 octets)                  |
+---------------------------------------+
```

NLRI Format: Route-type 1:

`[Type][Len][RD][ESI][ETag][MPLS Label]`

Net attributes: `[Type][RD][ESI][ETag]`

Path attributes: `[MPLS Label]`

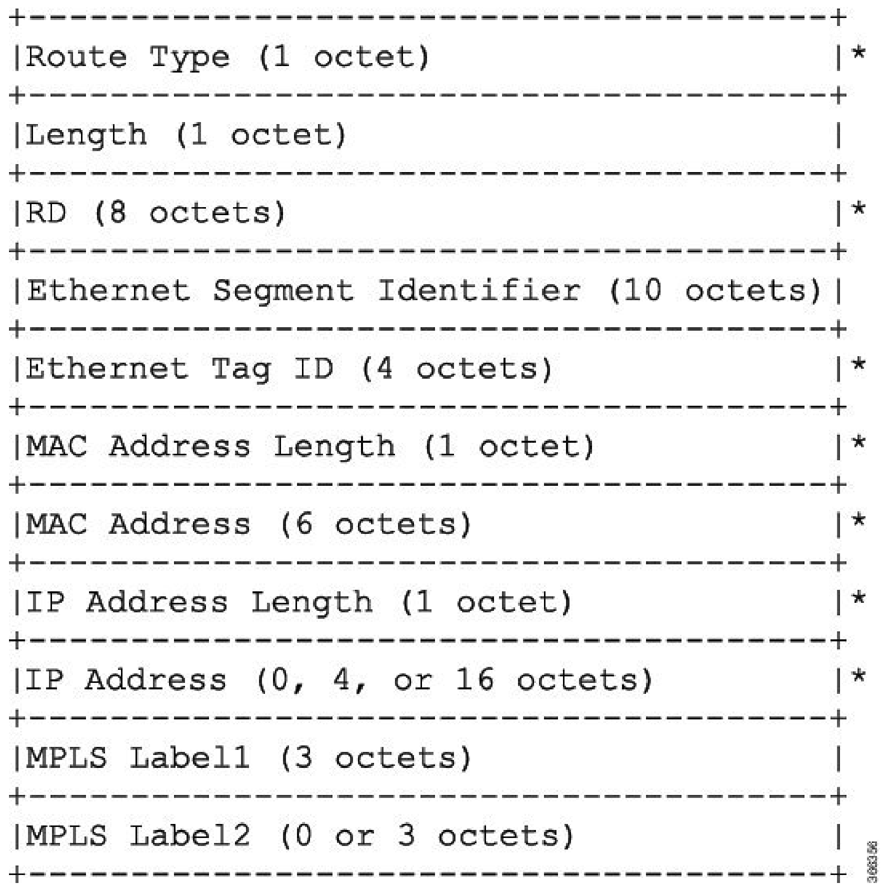
Example

```
route-policy evpn-policy
  if rd in (10.0.0.1:0) [and/or evpn-route-type is 1] [and/or esi in
(0a1.a2a3.a4a5.a6a7.a8a9)] [and/or etag is 4294967295] then
    set ..
  endif
end-policy
!
route-policy evpn-policy
  if rd  in (1.0.0.2:0) [and/or evpn-route-type is 1] [and/or esi in
(00a1.a2a3.a4a5.a6a7.a8a9)] [and/or etag is 4294967295] then
    set ..
  endif
end-policy
```

Route Type 2: MAC/IP Advertisement Route

The host's IP and MAC addresses are advertised to the peers within NLRI. The control plane learning of MAC addresses reduces unknown unicast flooding.

A MAC/IP Advertisement Route type specific EVPN NLRI consists of the following fields:



```
+---------------------------------------+
|Route Type (1 octet)                   |*
+---------------------------------------+
|Length (1 octet)                       |
+---------------------------------------+
|RD (8 octets)                          |*
+---------------------------------------+
|Ethernet Segment Identifier (10 octets)|
+---------------------------------------+
|Ethernet Tag ID (4 octets)             |*
+---------------------------------------+
|MAC Address Length (1 octet)           |*
+---------------------------------------+
|MAC Address (6 octets)                 |*
+---------------------------------------+
|IP Address Length (1 octet)            |*
+---------------------------------------+
|IP Address (0, 4, or 16 octets)        |*
+---------------------------------------+
|MPLS Label1 (3 octets)                 |
+---------------------------------------+
|MPLS Label2 (0 or 3 octets)            |
+---------------------------------------+
```

NLRI Format: Route-type 2:

`[Type][Len][RD][ESI][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr][MPLS Label1][MPLS Label2]`

Net attributes: `[Type][RD][ETag][MAC Addr Len][MAC Addr][IP Addr Len][IP Addr]`

Path attributes: `[ESI], [MPLS Label1], [MPLS Label2]`

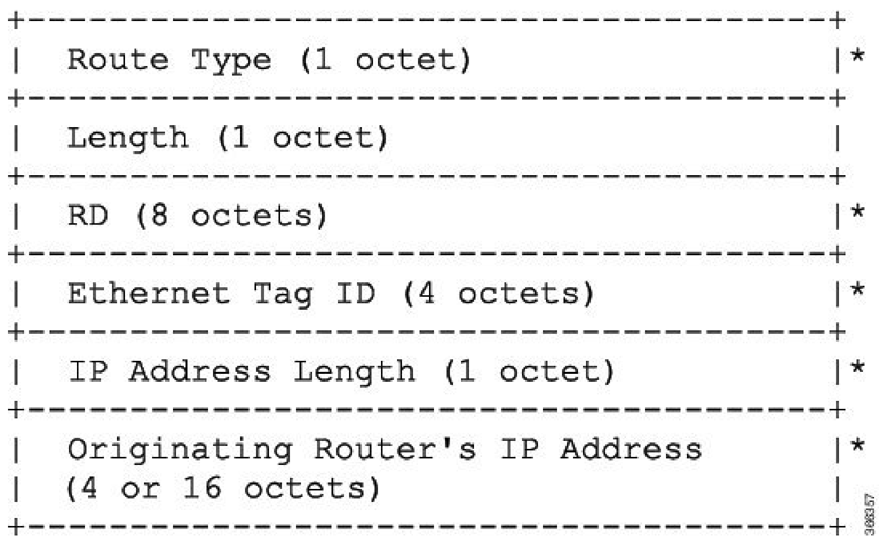
Example

```
route-policy evpn-policy
  if rd in (10.0.0.2:0) [and/or evpn-route-type is 2] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or macaddress in (0013.aabb.ccdd)]
[and/or destination in (1.2.3.4/32)] then
    set ..
  endif
end-policy
```

Route Type 3: Inclusive Multicast Ethernet Tag Route

This route establishes the connection for broadcast, unknown unicast, and multicast (BUM) traffic from a source PE to a remote PE. This route is advertised on per VLAN and per ESI basis.

An Inclusive Multicast Ethernet Tag route type specific EVPN NLRI consists of the following fields:


Route Type (1 octet) *
Length (1 octet)
RD (8 octets) *
Ethernet Tag ID (4 octets) *
IP Address Length (1 octet) *
Originating Router's IP Address (4 or 16 octets) *
368357

NLRI Format: Route-type 3:

```
[Type][Len][RD][ETag][IP Addr Len][Originating Router's IP Addr]
```

Net attributes: `[Type][RD][ETag][IP Addr Len][Originating Router's IP Addr]`

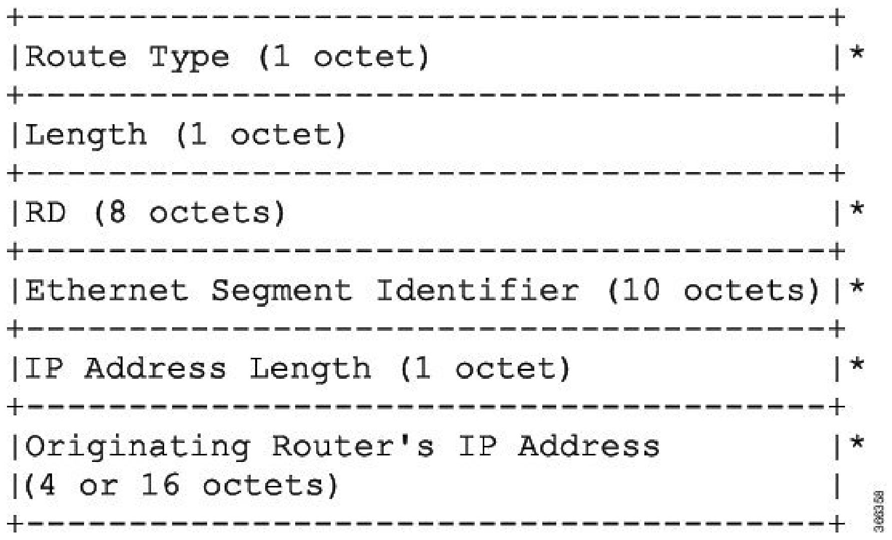
Example

```
route-policy evpn-policy
  if rd  in (10.0.0.1:300) [and/or evpn-route-type is 3] [and/or etag is 0] [and/or
evpn-originator in (10.0.0.1)] then
    set ..
  endif
end-policy
```

Route Type 4: Ethernet Segment Route

Ethernet segment routes enable to connect a CE device to two or PE devices. ES route enables the discovery of connected PE devices that are connected to the same Ethernet segment.

An Ethernet Segment route type specific EVPN NLRI consists of the following fields:



```
+---------------------------------------+
|Route Type (1 octet)                   |*
+---------------------------------------+
|Length (1 octet)                       |
+---------------------------------------+
|RD (8 octets)                          |*
+---------------------------------------+
|Ethernet Segment Identifier (10 octets)|*
+---------------------------------------+
|IP Address Length (1 octet)            |*
+---------------------------------------+
|Originating Router's IP Address        |*
|(4 or 16 octets)                       |
+---------------------------------------+
```

NLRI Format: Route-type 4:

`[Type][Len][RD][ESI][IP Addr Len][Originating Router's IP Addr]`

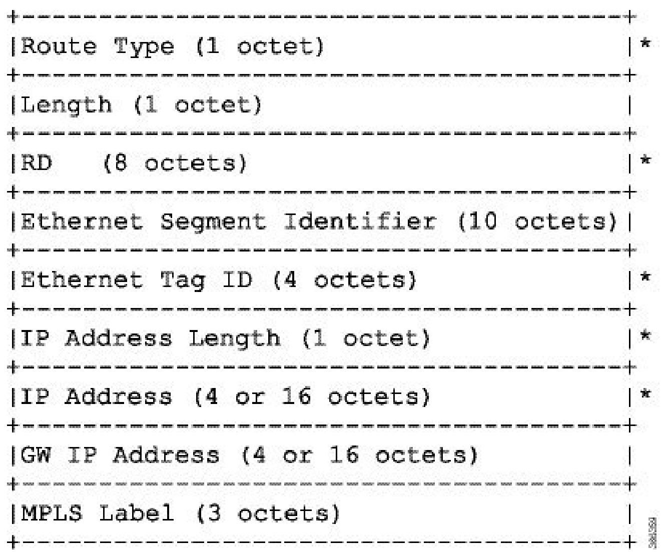
Net attributes: `[Type][RD][ESI][IP Addr Len][Originating Router's IP Addr]`

Example

```
route-policy evpn-policy
  if rd  in (10.0.0.1:0) [and/or evpn-route-type is 4] [and/or esi in
(00a1.a2a3.a4a5.a6a7.a8a9)] [and/or evpn-originator in (10.0.0.1)] then
    set ..
  endif
end-policy
```

Route Type 5: IP Prefix Route

An IP Prefix Route type specific EVPN NLRI consists of the following fields:



```
+---------------------------------------+
|Route Type (1 octet)                   |*
+---------------------------------------+
|Length (1 octet)                       |
+---------------------------------------+
|RD   (8 octets)                        |*
+---------------------------------------+
|Ethernet Segment Identifier (10 octets)|
+---------------------------------------+
|Ethernet Tag ID (4 octets)             |*
+---------------------------------------+
|IP Address Length (1 octet)            |*
+---------------------------------------+
|IP Address (4 or 16 octets)            |*
+---------------------------------------+
|GW IP Address (4 or 16 octets)         |
+---------------------------------------+
|MPLS Label (3 octets)                  |
+---------------------------------------+
```

NLRI Format: Route-type 5:

`[Type][Len][RD][ESI][ETag][IP Addr Len][IP Addr][GW IP Addr][Label]`

Net attributes: `[Type][RD][ETag][IP Addr Len][IP Addr]`

Path attributes: `[ESI], [GW IP Addr], [Label]`

Example

```
route-policy evpn-policy
  if rd in (30.30.30.30:1) [and/or evpn-route-type is 5] [and/or esi in
(0000.0000.0000.0000.0000)] [and/or etag is 0] [and/or destination in (12.2.0.0/16)] [and/or
 evpn-gateway in (0.0.0.0)] then
    set ..
  endif
end-policy
```

EVPN RPL Attribute

Route Distinguisher

A Route Distinguisher (rd) attribute consists of eight octets. An rd can be specified for each of the EVPN route types. This attribute is not mandatory in route-policy.

Example

```
rd in (1.2.3.4:0)
```

EVPN Route Type

EVPN route type attribute consists of one octet. This specifies the EVPN route type. The EVPN route type attribute is used to identify a specific EVPN NLRI prefix format. It is a net attribute in all EVPN route types.

Example

```
evpn-route-type is 3
```

```
The following are the various EVPN route types that can be used:
1 - ethernet-ad
2 - mac-advertisement
3 - inclusive-multicast
4 - ethernet-segment
5 – ip-advertisement
```

IP Prefix

An IP prefix attribute holds IPv4 or IPv6 prefix match specification, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. When IP prefix is specified in EVPN route type 2, it represents either a IPv4 or IPv6 host IP Address (/32 or /128). When IP prefix is specified in EVPN route type 5, it represents either IPv4 or IPv6 subnet. It is a net attribute in EVPN route type 2 and 5.

Example

```
destination in (128.47.10.2/32)
destination in (128.47.0.0/16)
destination in (128:47::1/128)
destination in (128:47::0/112)
```

esi

An Ethernet Segment Identifier (ESI) attribute consists of 10 octets. It is a net attribute in EVPN route type 1 and 4, and a path attribute in EVPN route type 2 and 5.

Example

```
esi in (ffff.ffff.ffff.ffff.fff0)
```

etag

An Ethernet tag attribute consists of four octets. An Ethernet tag identifies a particular broadcast domain, for example, a VLAN. An EVPN instance consists of one or more broadcast domains. It is a net attribute in EVPN route type 1, 2, 3 and 5.

Example

```
etag in (10000)
```

mac

The mac attribute consists of six octets. This attribute is a net attribute in EVPN route type 2.

Example

```
mac in (0206.acb1.e806)
```

evpn-originator

The evpn-originator attribute specifies the originating router's IP address (4 or 16 octets). This is a net attribute in EVPN route type 3 and 4.

Example

```
evpn-originator in (1.2.3.4)
```

evpn-gateway

The evpn-gateway attribute specifies the gateway IP address. The gateway IP address is a 32-bit or 128-bit field (IPv4 or IPv6), and encodes an overlay next-hop for the IP prefixes. The gateway IP address field can be zero if it is not used as an overlay next-hop. This is a path attribute in EVPN route type 5.

Example

```
evpn-gateway in (1.2.3.4)
```

EVPN RPL Attribute Set

In this context, the term set is used in its mathematical sense to mean an unordered collection of unique elements. The policy language provides sets as a container for groups of values for matching purposes. Sets are used in conditional expressions. The elements of the set are separated by commas. Null (empty) sets are allowed.

prefix-set

A prefix-set holds IPv4 or IPv6 prefix match specifications, each of which has four parts: an address, a mask length, a minimum matching length, and a maximum matching length. The address is required, but the other three parts are optional. The prefix-set specifies one or more IP prefixes.

Example

```
prefix-set ip_prefix_set
14.2.0.0/16,
54.0.0.0/16,
12.12.12.0/24,
50:50::1:0/112
end-set
```

mac-set

The mac-set specifies one or more MAC addresses.

Example

```
mac-set mac_address_set
1234.2345.6789,
2345.3456.7890
end-set
```

esi-set

The esi-set specifies one or more ESI's.

Example

```
esi-set evpn_esi_set
1234.2345.3456.4567.5678,
1234.2345.3456.4567.5670
end-set
```

etag-set

The etag-set specifies one or more Ethernet tags.

Example

```
etag-set evpn_etag_set
10000,
20000
end-set
```

EVPN Attributes and Operators

This table summarizes the EVPN attributes and operators per attach points.

Table 17: EVPN Attributes and Operators

Attach Point	Attribute	Match	Attribute-Set
neighbor-in	destination	in	—
	rd	in	—
	evpn-route-type	is	—
	esi	in	Yes
	etag	in	Yes
	mac	in	Yes
	evpn-originator	in	—
	evpn-gateway	in	—
neighbor-out	destination	in	—
	rd	in	—
	evpn-route-type	is	—
	esi	in	Yes
	etag	in	Yes
	mac	in	Yes
	evpn-originator	in	—
	evpn-gateway	in	—

Configure EVPN RPL Feature

The following section describe how to configure mac-set, esi-set, evpn-gateway, and evpn-originator.

```
/* Configuring a mac-set and refering it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# mac-set demo_mac_set
Router(config-mac)# 1234.ffff.aaa3,
Router(config-mac)# 2323.4444.ffff
Router(config-mac)# end-set
Router(config)# !
Router(config)# route-policy policy_use_pass_mac_set
Router(config-rpl)# if mac in demo_mac_set then
Router(config-rpl-if)# set med 200
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 1000
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# !
Router(config-bgp-af)# neighbor 10.0.0.10
Router(config-bgp-nbr)# remote-as 8
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy policy_use_pass_mac_set in
Router(config-bgp-nbr-af)# commit

/* Configuring a esi-set and refering it in a route-policy (Attach point - neighbor-in) */
Router# configure
Router(config)# esi-set demo_esi
Router(config-esi)# ad34.1233.1222.ffff.44ff,
Router(config-esi)# ad34.1233.1222.ffff.6666
Router(config-esi)# end-set
Router(config)# !
Router(config)# route-policy use_esi
Router(config-rpl)# if esi in demo_esi then
Router(config-rpl-if)# set local-preference 100
Router(config-rpl-if)# else
Router(config-rpl-else)# set local-preference 300
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit

/* Configuring evpn-gateway/evpn-originator in a route-policy (Attach point - neighbor-in
and out) */
Router# configure
Router(config)# route-policy gateway_demo
Router(config-rpl)# if evpn-gateway in (10.0.0.0/32) then
Router(config-rpl-if)# pass
Router(config-rpl-if)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# route-policy originator_demo
Router(config-rpl)# if evpn-originator in (10.0.0.1/32) then
Router(config-rpl-if)# set local-preference 100
Router(config-rpl-if)# else
Router(config-rpl-else)# set med 200
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy
Router(config)# commit
Router(config)# router bgp 100
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# !
Router(config-bgp-af)# neighbor 10.0.0.10
Router(config-bgp-nbr)# remote-as 8
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy gateway_demo in
Router(config-bgp-nbr-af)# route-policy originator_demo out
Router(config-bgp-nbr-af)# commit
```

Running Configuration

```
/* Configuring a mac-set and refering it in a route-policy (Attach point - neighbor-in) */
mac-set demo_mac_set
  1234.ffff.aaa3,
  2323.4444.ffff
end-set
!
route-policy policy_use_pass_mac_set
  if mac in demo_mac_set then
```

```
    set med 200
  else
    set med 1000
  endif
end-policy
!
router bgp 100
 address-family l2vpn evpn
 !
 neighbor 10.0.0.10
  remote-as 8
  address-family l2vpn evpn
  route-policy policy_use_pass_mac_set in
  !
 !
!
end

/* Configuring a esi-set and refering it in a route-policy (Attach point - neighbor-in) */
Wed Oct 26 11:52:23.720 IST
esi-set demo_esi
  ad34.1233.1222.ffff.44ff,
  ad34.1233.1222.ffff.6666
end-set
!
route-policy use_esi
  if esi in demo_esi then
    set local-preference 100
  else
    set local-preference 300
  endif
end-policy
```

EVPN Route Policy Examples

```
route-policy ex_2
  if rd in (2.2.18.2:1004) and evpn-route-type is 1 then
    drop
  elseif rd in (2.2.18.2:1009) and evpn-route-type is 1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy ex_3
  if evpn-route-type is 5 then
    set extcommunity bandwidth (100:9999)
  else
    pass
  endif
end-policy
!
route-policy samp
end-policy
!
route-policy samp1
  if rd in (30.0.101.2:0) then
    pass
  endif
end-policy
!
```

```
route-policy samp2
  if rd in (30.0.101.2:0, 1:1) then
    pass
  endif
end-policy
!
route-policy samp3
  if rd in (*:*) then
    pass
  endif
end-policy
!
route-policy samp4
  if rd in (30.0.101.2:*) then
    pass
  endif
end-policy
!
route-policy samp5
  if evpn-route-type is 1 then
    pass
  endif
end-policy
!
route-policy samp6
  if evpn-route-type is 2 or evpn-route-type is 5 then
    pass
  endif
end-policy
!
route-policy samp7
  if evpn-route-type is 4 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp8
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 then
    pass
  endif
end-policy
!
route-policy samp9
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
 is 4 then
    pass
  endif
end-policy
!
route-policy test1
  if evpn-route-type is 2 then
    set next-hop 10.2.3.4
  else
    pass
  endif
end-policy
!
route-policy test2
  if evpn-route-type is 2 then
    set next-hop 10.10.10.10
  else
    drop
  endif
end-policy
```

```
!
route-policy test3
  if evpn-route-type is 1 then
    set tag 9988
  else
    pass
  endif
end-policy
!
route-policy samp21
  if mac in (6000.6000.6000) then
    pass
  endif
end-policy
!
route-policy samp22
  if extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp23
  if evpn-route-type is 1 and esi in (aaaa.bbbb.cccc.dddd.eeee) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp24
  if evpn-route-type is 5 and extcommunity rt matches-any (100:1001) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp25
  if evpn-route-type is 2 and esi in (1234.1234.1234.1234.1236) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp26
  if etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp27
  if destination in (99.99.99.1) and etag in (20000) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp31
```

```
  if evpn-route-type is 1 or evpn-route-type is 2 or evpn-route-type is 3 or evpn-route-type
 is 4 or evpn-route-type is 5 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp33
  if esi in evpn_esi_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp34
  if destination in (90:1:1::9/128) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp35
  if destination in evpn_prefix_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp36
  if evpn-route-type is 3 and evpn-originator in (80:1:1::3) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp37
  if evpn-gateway in (10:10::10) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp38
  if mac in evpn_mac_set1 then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp39
  if mac in (6000.6000.6002) then
    pass
  else
    drop
  endif
end-policy
!
```

```
route-policy samp41
  if evpn-gateway in (10.10.10.10, 10:10::10) then
    pass
  else
    drop
  endif
end-policy
!
route-policy samp42
  if evpn-originator in (24.162.160.1/32, 70:1:1::1/128) then
    pass
  else
    drop
  endif
end-policy
!
route-policy example
  if rd in (62300:1903) and evpn-route-type is 1 then
    drop
  elseif rd in (62300:19032) and evpn-route-type is 1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp100
  if evpn-route-type is 4 or evpn-route-type is 5 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp101
  if evpn-route-type is 4 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp102
  if evpn-route-type is 4 then
    drop
  elseif evpn-route-type is 5 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp103
  if evpn-route-type is 2 and destination in evpn_prefix_set1 then
    drop
  else
    pass
  endif
end-policy
!
route-policy samp104
  if evpn-route-type is 1 and etag in evpn_etag_set1 then
    drop
  elseif evpn-route-type is 2 and mac in evpn_mac_set1 then
```

```
    drop
  elseif evpn-route-type is 5 and esi in evpn_esi_set1 then
    drop
  else
    pass
  endif
end-policy
!
```

BGP Multiple Sourced or Redistributed Paths

The BGP Multiple Sourced or Redistributed Paths feature allows BGP to receive multiple paths for each prefix that is redistributed or locally sourced. These multipaths can be used for add-path functionality advertisement. This feature allows the Virtual Topology System (VTS) to advertise the routes along with its IP address even when the Virtual Traffic Forwarder (VTF) resides outside the VTS controller. This enables the VTS customers to use multipath load-balancing capabilities across multiple VTFs.

The VTS advertises multiple paths of its VTFs to the remote autonomous system add path along with the properties of its own path, such as load-metrics and VXLAN Network Identifier (VNIs). The VTS uses the Server Layer applications for this advertisement. This enables multipath capability across VTFs along with load balancing.

Configure BGP Multiple Sourced or Redistributed Paths

You can configure the BGP Multiple Sourced or Redistributed Paths feature for redistributed or locally sourced prefix.

Perform the following tasks to configure BGP Multipath Extensions for redistributed prefix.

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# vrf vrf-1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# redistribute application Service-layer multipath
Router(config-bgp-vrf-af)# commit
```

Perform the following tasks to configure BGP Multipath Extensions for locally sourced prefix.

```
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# vrf vrf-1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# network 192.0.2.1 255.255.255.0 multipath
Router(config-bgp-vrf-af)# commit
```

Running Configuration

This section shows BGP Multiple Sourced and Redistributed Paths running configuration.

```
/* For redistributed prefix */
configure
 router bgp 100
  vrf vrf-1
    address-family ipv4 unicast
      redistribute application Service-layer multipath
```

```
!
!

/* For locally sourced prefix */
configure
 router bgp 200
  vrf vrf-1
    address-family ipv4 unicast
      network 192.0.2.1 255.255.255.0 multipath
!
!
```

Verification

Verify the BGP Multiple Sourced or Redistributed Paths feature configuration.

```
Router# show bgp vrf vrf-1 198.51.100.1/32
Fri Nov 16 19:03:08.727 PST
BGP routing table entry for 198.51.100.1/32, Route Distinguisher: 192.168.0.1:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 10          10
    Local Label: 24001
Last Modified: Nov 16 15:47:24.000 for 03:15:45
Paths: (2 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (192.168.0.1)
      Received Label 10000
      Origin incomplete, metric 5, localpref 100, weight 32768, valid, redistributed, best,
 group-best, import-candidate
      Received Path ID 1, Local Path ID 1, version 10
      Extended community: Encapsulation Type:8 Router MAC:abcd.ef00.0101 RT:10:10
     VPN Nexthop: 10.0.0.1 <------------------------------------------------------------
 PATH1
  Path #2: Received by speaker 0
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (192.168.0.1)
      Received Label 10000
      Origin incomplete, metric 5, localpref 100, weight 32768, valid, redistributed,
add-path
      Received Path ID 2, Local Path ID 2, version 10
      Extended community: Encapsulation Type:8 Router MAC:abcd.ef00.0102 RT:10:10
     VPN Nexthop: 10.0.0.2 <------------------------------------------------------------
 PATH2
```

Related Topics

- #unique_663

Associated Commands

- redistribute application Service-layer multipath
- network <ip address> multipath
- show bgp vrf <vrf_name>

Highest Random Weight Mode for EVPN DF Election

The Highest Random Weight (HRW) Mode for EVPN DF Election feature provides optimal load distribution of Designated Forwarder (DF) election, redundancy, and fast access. It ensures a nondisruptive service for an ES irrespective of the state of a peer DF.

The DF election is calculated based on the weight. The highest weight becomes the DF and the subsequent weight becomes a backup DF (BDF). The weight is determined by the mathematical function of EVI, ESI, and the IP address of the server.

DF weight calculation is based on the weight vector:

```
Wrand(v, Si) = (1103515245((1103515245.Si+12345)XOR
       D(v))+12345)(mod 2^31)
                       where:
             Si: IP Address of the server i
             v: EVI
             D(v): 31 bit digest [CRC-32 of v]
```

The existing DF election algorithm is based on ordinal value of a modulus calculation, and it comprises of number of peers and EVI. The DF is determined by the mathematical function of ESI and EVI, which is called “service carving”. This mode of DF election is described in RFC 7432.

In modulus calculation mode, the algorithm does not perform well when the Ethernet tags are all even or all odd. When the Ethernet Segment (ES) is multihomed to two PEs, all the VLANs pick only one of the PEs as the DF; one of the PEs does not get elected at all as the DF. The DF election is not optimal in this mode of operation.

The HRW mode of DF election has the following advantages over modulus mode of DF election:

- The DF election for the respective VLANs is equally distributed among the PEs.
- When a PE which is neither a DF nor a BDF hosts some VLANs on a given ES, and if the PE goes down, or its connection to the ES goes down, it does not result in a DF and BDF reassignment to the other PEs. This eliminates computation during the connection flaps.
- It avoids the service disruption that are inherent in the existing modulus based algorithm.
- The BDF provides redundant connectivity. The BDF ensures that there is no traffic disruption when a DF fails. When a DF fails, the BDF becomes the DF.

Configure Highest Random Weight Mode for EVPN DF Election

Perform this task to configure Highest Random Weight Mode for EVPN DF Election feature.

Configuration Example

```
Router# configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether 23
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#service-carving hrw
Router(config-evpn-ac-es)#commit
```

Running Configuration

```
configure
 evpn
  interface Bundle-Ether 23
   ethernet-segment
    service-carving hrw
   !
  !
 !
```

Verification

Verify that you have configured HRW mode of DF election.

```
Router#show evpn ethernet-segment interface bundleEther 23 carving detail
Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0011.1111.1111.1111.1111 Gi0/2/0/0                          192.168.0.2
                                                            192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : GigabitEthernet0/2/0/0
     Interface MAC  : 02db.c740.ca4e
     IfHandle       : 0x01000060
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 11.1111.1111.1111.1111
  ES Import RT      : 0011.0011.0011 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS) (default)
  Service Carving   : HRW    -> Operation mode of carving
  Peering Details   : 192.168.0.2[HRW:P:00] 192.168.0.3[HRW:P:00] -> Carving capability as
 advertised by peers
  Service Carving Results:
     Forwarders     : 1
     Permanent      : 0
     Elected        : 0
     Not Elected    : 1
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 28109
  Remote SHG labels : 1
              24016 : nexthop 192.168.0.3
```

Associated Commands

- service-carving
- show evpn ethernet-segment

Layer 2 Fast Reroute

Table 18: Feature History Table

Feature Name	Release Information	Feature Description
Layer 2 Fast Reroute	Release 7.3.1	In the event of a link failure, this feature enables the router to switch traffic quickly to a precomputed loop-free alternative (LFA) path by allocating a label to the incoming traffic. This minimizes the traffic loss ensuring fast convergence. This feature introduces the **convergence reroute** command.

When there is a link failure, a network experiences traffic loss for a brief period until the convergence is complete. The extent of traffic loss depends on various factors such as the performance of the control plane, tuning of fast convergence, and the choice of technologies of the control plane on each node in the network.

Certain fault-tolerant applications are impacted by the traffic loss. To reduce this traffic loss, a technique for data plane convergence is essential. Fast Reroute (FRR) is one such technique that is primarily applicable to the network core.

The Layer 2 Fast Reroute (L2 FRR) feature enables the router to quickly send the traffic through the backup path when a primary link fails. The feature helps to minimise traffic loss and ensures fast convergence.

L2 FRR precomputes the loop-free alternative (LFA) path in the hardware. When a link or a router fails, distributed routing algorithms takes the failure into account and compute new routes. The time taken for computation is called routing transition. The routing transition in BGP convergence can take up to several hundreds of milliseconds.

Use LFA FRR to reduce the routing transition time to less than 50 milliseconds using a precomputed alternate backup path. When a router detects a link failure, FRR allocates a label to the incoming traffic, and the router immediately switches the traffic over to the backup path to reduce traffic loss.

Benefits

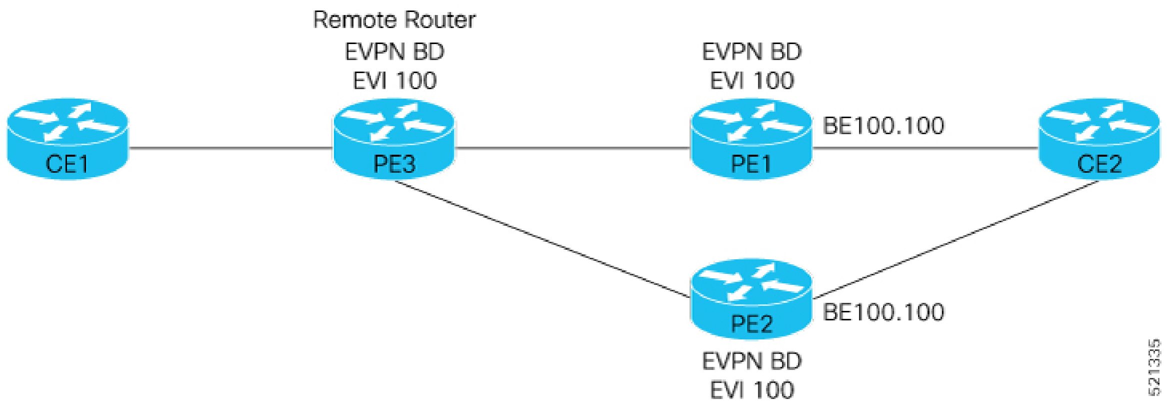
This feature provides fast and predictable convergence:

- Convergence time is 50 ms
- Fast failure notification even in large rings with high number of nodes
- Manual configuration for predictable failover behavior
- You do not have to change the topology

Restrictions

- You can use L2 FRR only when PE devices are in EVPN active-active or single-active mode.
- L2 FRR is applicable only for unicast traffic and not for BUM traffic.

Figure 94: Layer 2 Fast Reroute



In this topology:

- CE2 is multihomed to PE1 and PE2.
- PE1 and PE2 are in EVPN active-active or single-active mode. They are connected to a remote router PE3 over the MPLS core network.
- CE1 is connected to PE3.
- Both PE1 and PE2 are L2 FRR enabled. An FRR label is added per EVI for the backup path.

Consider a traffic flow from CE1 to CE2 in a regular scenario:

- The traffic is sent from CE1 to PE3.
- PE3 distributes the traffic over PE1 and PE2.
- PE1 and PE2 sends the traffic to CE2.

When FRR is enabled:

- When the PE1-CE2 link goes down, L2 FRR is triggered on PE1. Traffic is redirected to PE2 until the convergence is complete.
- When you enable FRR on PE1, the logical backup path is pre-programmed in the hardware. When PE1 detects a failure on the access side (CE2), PE1 identifies the backup PE2 as has been programmed in the hardware.
- PE1 allocates the FRR label to the incoming traffic to reach PE2.
- All incoming traffic to PE1 is redirected to PE2 using this FRR label.
- PE1 encapsulates all the traffic with the label of PE2 and forwards the traffic to PE2.
- PE2 receives the traffic with the label.
- Each interface has an unique label.
- PE2 removes the FRR label and forwards the traffic to the correct AC.

Configure Layer 2 Fast Reroute

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether100
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.00.00.00.00.00.05.01.02
Router(config-evpn-ac-es)# convergence
Router(config-evpn-ac-es-conv)# reroute
Router(config-evpn-ac-es-conv)# bgp route-target 5000.5000.5002
Router(config-evpn-ac-es-conv)# exit
Router(config-evpn-ac-es)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# interface Bundle-Ether100.100 > L2FRR enabled bridge-port (BP),
 Primary and backup learn-fecs will be created in the hardware for this BP
Router(config-l2vpn-bg-bd-ac)# routed interface BVI100
Router(config-l2vpn-bg-bd-bvi)# evi 990
Router(config-l2vpn-bg-bd-evi)# commit
```

Running Configuration

This section shows the Layer 2 Fast Reroute running configuration.

```
evpn
 interface Bundle-Ether100
  ethernet-segment
   identifier type 0 00.00.00.00.00.00.05.01.02
   convergence
    reroute
   !
   bgp route-target 5000.5000.5002
  !
 !
!
l2vpn
 bridge group bg1
  bridge-domain bd1
   interface Bundle-Ether100.100
   !
   routed interface BVI100
   !
   evi 990
   !
  !
 !
!
end
```

Verification

Verify that you have configured Layer 2 Fast Reroute successfully.Check ESI bundle carving details, and ensure convergence reroute is enabled.

```
Router:PE1#show evpn ethernet-segment interface bundle-Ether 100 carving details
Mon Dec  7 11:04:37.604 UTC
Legend:
  B   - No Forwarders EVPN-enabled,
  C   - Backbone Source MAC missing (PBB-EVPN),
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated

Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0000.0000.0000.0005.0102 BE200                              2.2.2.2
                                                            8.8.8.8
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether200
     Interface MAC  : 008a.960e.70d9
     IfHandle       : 0x20004064
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 00.0000.0000.0005.0102
  ES Import RT      : 5000.5000.5002 (Local)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, All-active
     Configured     : All-active (AApF) (default)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       : Reroute, <<<<<<<< Check reroute is enabled on this ESI bundle
     Mobility-Flush : Debounce 1 sec, Count 0, Skip 0
                    : Last n/a
  Peering Details   : 2 Nexthops
     2.2.2.2 [MOD:P:7fff:T]
     8.8.8.8 [MOD:P:00:T]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
  Service Carving Results:
     Forwarders     : 2
     Elected        : 1
            EVI E   :      990
     Not Elected    : 1
            EVI NE  :      991
  EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  Local SHG label   : 28098
  Remote SHG labels : 1
              28098 : nexthop 8.8.8.8
  Access signal mode: Bundle OOS (Default)
```

Check multihoming nodes per bridge-port (BP) AC backup information is programmed correctly.

```
Router:PE1#show l2vpn forwarding interface bundle-Ether100.100 private location 0/0/CPU0
```

```
Mon Dec  7 11:04:37.604 UTC

Xconnect ID 0xa0000007

  Xconnect info:
   Base info: version=0xaabbcc13, flags=0xc110, type=2, reserved=0, address=0x308b25c448
    xcon_bound=TRUE, switching_type=0, data_type=12
    xcon_name=

  AC info:
   Base info: version=0xaabbcc11, flags=0x0, type=3, reserved=0, address=0x308b25c570
    xcon_id=0xa0000007, ifh=0x2000403c, subifh=0x20004046, ac_id=0, ac_type=21,
    ac_mtu=1500, iw_mode=1, adj_valid=TRUE, adj_addr=0x20004046, adj_ptr=0x30887ab5c8
    r_aps_channel=FALSE, prot_exclusion=FALSE
    rg_id=0, ro_id=0x0000000000000000
    evpn internal label = None
    E-Tree = Root
    FXC local-switch AC xcid = 0x0 (Invalid)
    FXC local-switch PW xcid = 0xffffffff (Invalid)
    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
      packets dropped: PLU 0, tail 0
      bytes dropped: PLU 0, tail 0

  AC Backup info:
   Base info: version=0xaabbcc39, flags=0x0, type=43, reserved=0, address=0x308b264b70
      VC label: 28100  << FRR label advertised by remote multihome peer node. Check this
label details on the multihoming peer node is correct or not. Note down this label and check
 on MH2 node.

   Object: AC_BACKUP
   Base info: version=0xaabbcc39, flags=0x0, type=43, reserved=0, address=0x308b264b70
   Event Trace History [Total events: 1]
--------------------------------------------------------------------------------
     Time               Event               Flags
     ====               =====               =====
     Jul 23 08:59:49.658 Create             0x0          -  -
--------------------------------------------------------------------------------

  Nexthop info:
   Base info: version=0xaabbcc14, flags=0x10000, type=7, reserved=0, address=0x308b25b4f8
    nh_addr=2.2.2.2
    ecd_plat_data_valid=TRUE, ecd_plat_data_len=104, plat_data_size=496
    child_count=0, child_evpn_ole_count=2, child_mac_count=0, child_pwhe_mp_count=0,
child_ac_backup_count=2

   Object: NHOP
   Base info: version=0xaabbcc14, flags=0x10000, type=7, reserved=0, address=0x308b25b4f8
   Event Trace History [Total events: 5]
--------------------------------------------------------------------------------
     Time               Event               Flags
     ====               =====               =====
     Jul 16 17:01:54.093 Create             0x0          -  -
     Jul 16 17:01:54.093 FIB reg            0x2300       -  -
     Jul 16 17:01:54.093 PD create          0x300        -  -
     Jul 16 17:01:54.093 FIB ecd            0x200        -  S
     Jul 16 17:01:54.093 PD bind            0x10200      -  -
--------------------------------------------------------------------------------

  Bridge port info:
   Base info: version=0xaabbcc1a, flags=0x0, type=12, reserved=0, address=0x308b25c690
```

```
    xcon_id=0xa0000007, bridge_id=0, shg_id=0, mac_limit=64000, mac_limit_action=0,
    bridge_ptr=0x8b263f68, shg_ptr=0x0, msti_ptr=0x8b2613a0, segment_ptr=0x8b25c570
    segment_type=0x2, mtu=1500, msti=7, mvrp_seq_number=0, learn_key = 0
    is_flooding_disabled=FALSE, is_mac_learning_disabled=FALSE,
is_mac_port_down_flush_disabled=FALSE, mtu=1500, msti=7,
    aging_timeout=300, bridge_ptr=0x8b263f68, shg_ptr=0x0, segment_type=2,
    segment_ptr=0x8b25c570
    MAC learning: enabled
    Software MAC learning: enabled
    MAC port down flush: enabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown unicast: enabled
    MAC aging time: 300 s, Type: inactivity
    MAC limit: 64000, Action: none, Notification: syslog
    MAC limit reached: no, threshold: 75%
    MAC Secure: disabled, Logging: disabled, Accept-Shutdown: enabled
    DHCPv4 snooping: profile not known on this node, disabled
    Dynamic ARP Inspection: disabled, Logging: disabled
    IP Source Guard: disabled, Logging: disabled
    IGMP snooping profile: profile not known on this node
    MLD snooping profile: profile not known on this node
    Router guard disabled
    STP participating: disabled
    Storm control: disabled
    PD System Data: AF-LIF-IPv4: 0x00014003  AF-LIF-IPv6: 0x00014004 FRR-LIF: 0x00014002


   Object: XCON
   Base info: version=0xaabbcc13, flags=0xc110, type=2, reserved=0, address=0x308b25c448
   Event Trace History [Total events: 14]
--------------------------------------------------------------------------------
     Time                Event                Flags
     ====                =====                =====
     Jul 23 08:59:48.126 Create               0x80000      -  -
     Jul 23 08:59:48.126 Modify_RED           0x113        M  -
     Jul 23 08:59:48.126 PD create            0x100110     -  -
     Jul 23 08:59:48.126 PD create            0x2110       -  -  (seg2)
     Jul 23 08:59:48.126 PD bind              0x110        -  -
     Jul 23 08:59:48.126 PD bind              0x2110       -  -  (seg2)
     Jul 23 08:59:48.126 Modify_RED           0x113        M  -
     Jul 23 08:59:48.126 DHCP upd             0xc110       -  -
     Jul 23 08:59:48.126 AIB upd              0xc110       -  -
     Jul 23 08:59:48.126 PD modify            0x110        -  -
     Jul 23 08:59:49.662 Modify               0x100110     -  -
     Jul 23 08:59:49.662 Modify_RED           0x110        -  -
     Jul 23 08:59:49.662 PD modify            0x110        -  -
     Jul 23 08:59:49.662 DHCP upd             0xc110       -  -
--------------------------------------------------------------------------------
   Modify Event History, oldest - 0x0 - 0x100110
RP/0/RP0/CPU0:shan-evpn-leaf1#

/* Check MH2 node FRR-label 28100 output */
Router:PE#show mpls forwarding labels 28100
Mon Dec  7 11:04:37.604 UTCv
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
28100  Pop         PW(127.0.0.1:4254701977609)    \
                                      BE200.200    point2point     0
```

Associated Commands

- **convergence reroute**

- **show evpn ethernet-segment**
- **show evpn evi**
- **show evpn evi ead private**

EVPN Preferred Nexthop

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Preferred Nexthop	Release 7.3.1	With this feature, you can set an active and backup path, in a dual-homed mode based on the nexthop IP address, thereby allowing greater control over traffic patterns. If you are unable to use single-active mode due to hardware, topology, or technological limitations, this feature enables you to direct traffic to a specific remote PE. This feature introduces the **preferred nexthop** command.

The EVPN Preferred Nexthop feature allows you to choose a primary nexthop and backup nexthop among the remote PE devices in dual-homed mode. By default, in an all-active dual-homed topology, traffic is load balanced using ECMP across both remote PE devices.

Configure the **preferred-nexthop** command when you want to direct traffic to one specific remote PE, and you are unable to use single-active mode due to hardware, topology, or technological limitations. The router allocates an internal label and will not allocate or consume ECMP FEC. The internal label enables fast switchover to backup PE when the primary link fails.

When remote PEs are operating in EVPN all-active mode, configure the **preferred-nexthop** command per EVI to choose an active and backup path based on the nexthop IP address. You can set the highest IP address as primary, which results in the lower IP address as a backup or vice versa.This feature provides you greater control over traffic patterns, that is to achieve symmetric traffic flow, and to allow support when a topology cannot support an all-active remote PE. Preferred nexthop is supported for native EVPN, EVPN VPWS, and EVPN PWHE. This feature supports a topology that has only two remote nexthops.

Configure EVPN Preferred Nexthop

Perform the following task to configure EVPN preferred nexthop.

Configuration Example

This example shows the configuration of highest IP address as the preferred nexthop.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-evi)# preferred-nexthop highest-ip
Router(config-evpn-evi)# commit
```

This example shows the configuration of lowest IP address as the preferred nexthop.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-evi)# preferred-nexthop lowest-ip
Router(config-evpn-evi)# commit
```

This example shows the configuration of preferred nexthop using the **modulo** keyword.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 100
Router(config-evpn-evi)# preferred-nexthop modulo
Router(config-evpn-evi)# commit
```

Running Configuration

This section shows the EVPN preferred nexthop running configuration.

```
/* Configuration of highest IP address as the preferred nexthop */
evpn
 evi 100
  preferred-nexthop highest-ip
!

/* Configuration of lowest IP address as the preferred nexthop */
evpn
 evi 100
  preferred-nexthop lowest-ip
!

/* Configuration of preferred nexthop using the modulo keyword */
evpn
 evi 100
  preferred-nexthop modulo
```

Verification

The output shows that the Highest IP is selected as primary (P) and the lowest IP as backup (B). The path selection is programmed in CEF.

```
Router#show evpn evi vpn-id 100 detail
Mon Oct 26 14:00:51.459 EDT

VPN-ID     Encap      Bridge Domain                Type
---------- ---------- ---------------------------- -------------------
100        MPLS       bd100                        EVPN
…
   Preferred Nexthop Mode: Highest IP

Router#show evpn internal-label vpn-id  100 detail
Mon Oct 26 14:01:46.665 EDT

VPN-ID     Encap  Ethernet Segment Id         EtherTag     Label
---------- ------ --------------------------- ----------   --------
100        MPLS   0100.0000.acce.5500.0100    0            28120
      Multi-paths resolved: TRUE (Remote all-active) (Preferred NH, Highest IP)
      Multi-paths Internal label: 28120
      EAD/ES      192.168.0.1                              0
                            192.168.0.3                              0
      EAD/EVI     192.168.0.1                              28099
                        192.168.0.3                             28099
      Summary pathlist:
     0xffffffff (P) 192.168.0.3                              28099
      0xffffffff (B) 192.168.0.1                              28099

Router#show cef mpls local-label 28120 eOS
Mon Oct 26 14:04:10.851 EDT
Label/EOS 28120/1, version 56, internal 0x1000001 0x30 (ptr 0x4d3ba2a8) [1], 0x0 (0x0),
0x208 (0x4e6502c0)
Updated Oct 26 14:00:31.225
…
   via 192.168.0.3/32, 6 dependencies, recursive [flags 0x0]
    path-idx 0 NHID 0x0 [0x4d3bb58c 0x0], Internal 0x4e7890f8
    recursion-via-/32
    next hop 192.168.0.3/32 via 28103/0/21
     local label 28120
     next hop 27.27.27.3/32 Gi0/2/0/7    labels imposed {ImplNull 28099}
   via 192.168.0.1/32, 6 dependencies, recursive, backup (Local-LFA) [flags 0x300]
    path-idx 1 NHID 0x0 [0x4d3bb454 0x0]
    recursion-via-/32
    next hop 192.168.0.1/32 via 28105/0/21
     local label 28120
     next hop 26.26.26.1/32 Gi0/2/0/6    labels imposed {ImplNull 28099}
```

EVPN Access-Driven DF Election

Table 20: Feature History Table

Feature Name	Release Information	Feature Description
EVPN Access-Driven DF Election	Release 7.3.1	This feature enables the access network to control EVPN PE devices by defining the backup path much before the event of a link failure, thereby reducing the traffic loss. The following keywords are added to the **service-carving** command: • **preference-based** • **access-driven**

This feature includes a preference-based and access-driven DF election mechanism.

In a preference-based DF election mechanism, the weight decides which PE is the DF at any given time. You can use this method for topologies where interface failures are revertive. However, for topologies where an access-PE is directly connected to the core PE, use the access-driven DF election mechanism.

When access PEs are configured in a non-revertive mode, the access-driven DF election mechanism allows the access-PE to choose which PE is the DF.

Consider an interface in an access network that connects PE nodes running Multichassis Link Aggregation Control Protocol (mLACP) and the EVPN PE in the core. When this interface fails, there may be a traffic loss for a longer duration. The delay in convergence is because the backup PE is not chosen before failure occurs.

The EVPN Access-Driven DF Election feature allows the EVPN PE to preprogram a backup PE even before the failure of the interface. In the event of failure, the PE node will be aware of the next PE that will take over. Thereby reducing the convergence time. Use the *preference df weight* option for an Ethernet segment identifier (ESI) to set the backup path. By configuring the weight for a PE, you can control the DF election, thus define the backup path.

Restrictions

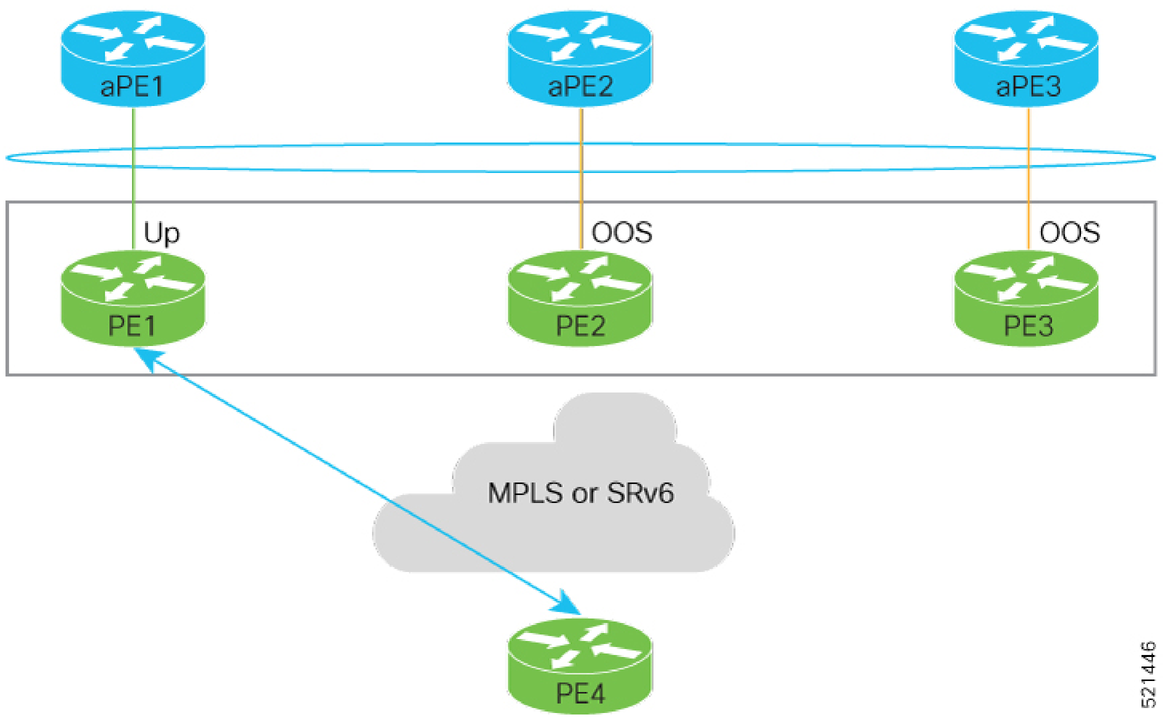
- The feature is supported only in an EVPN-VPWS scenario where EVPN PEs are in the port-active mode.
- The bundle attached to the ethernet segment must be configured with **lacp mode active**.

  LACP mode on is not supported.

Topology

Let's understand the feature on how the backup path is precomputed with the following topology.

Figure 95: EVPN Access-Driven DF Election

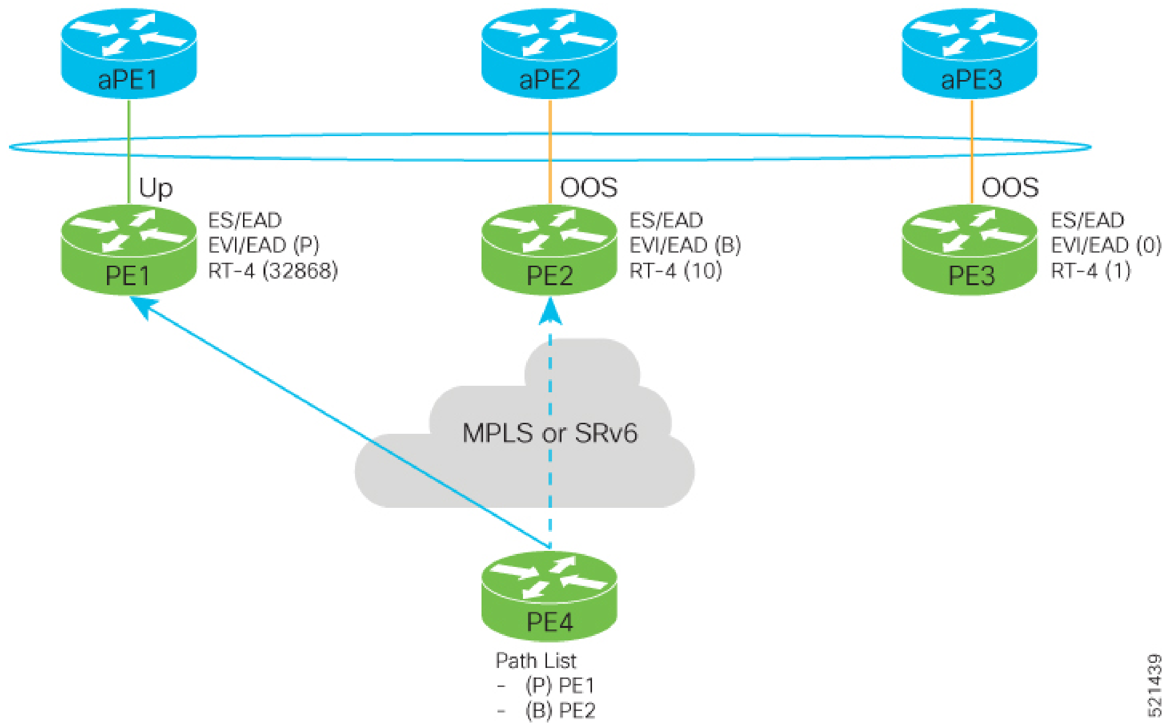


- PE1, PE2, and PE3 are PEs for the EVPN core network.
- aPE1, aPE2, and aPE3 are their access PE counterparts and configured in a multichassis link aggregation group (MCLAG) redundancy group. Only one link among the three is active at any given time. aPE1, aPE2, and aPE3 are in a non-revertive mode.
- PE1 is directly connected to aPE1, PE2 to aPE2, and PE3 to aPE3. EVPN VPWS is configured on the PE devices in the core.
- All PE devices are attached to the same bundle and shares the same ethernet segment identifier.

- PE1, PE2, and PE3 are configured with a weight of 100, 10, and 1 respectively.

Traffic Flow

In this example, consider a traffic flow from a host connected to PE4 to the host connected to the access PE.



- aPE1-PE1 interface state is up. The aPE2-PE2 and aPE3-PE3 remains in OOS state.
- The traffic is sent from PE4 to aPE1 through PE1 as the PE1 is configured with a highest weight of 100.
- The highest weight is modified by adding 32768 to the configured weight. For example, the weight of PE1 is 100, 32768 is added to this weight. Hence, 32868 is advertised to the peer PEs.
- The highest weight is advertised as P-bit, which is primary. The next highest weight is advertised as B-bit, which is secondary. The lowest weight as non-DF (NDF).
- When the EVPN PE devcies are of same weight, the traffic is sent based on the IP address. Lowest IP address takes the precedence.
- Only one PE indicates that the state of the bundle for the Ethernet Segment is up. For all other PEs, the Ethernet Segment is standby and the bundle is in OOS state.
- All PE devices are aware of the associated next hop and weights of their peers.

Failure and Recovery Scenarios

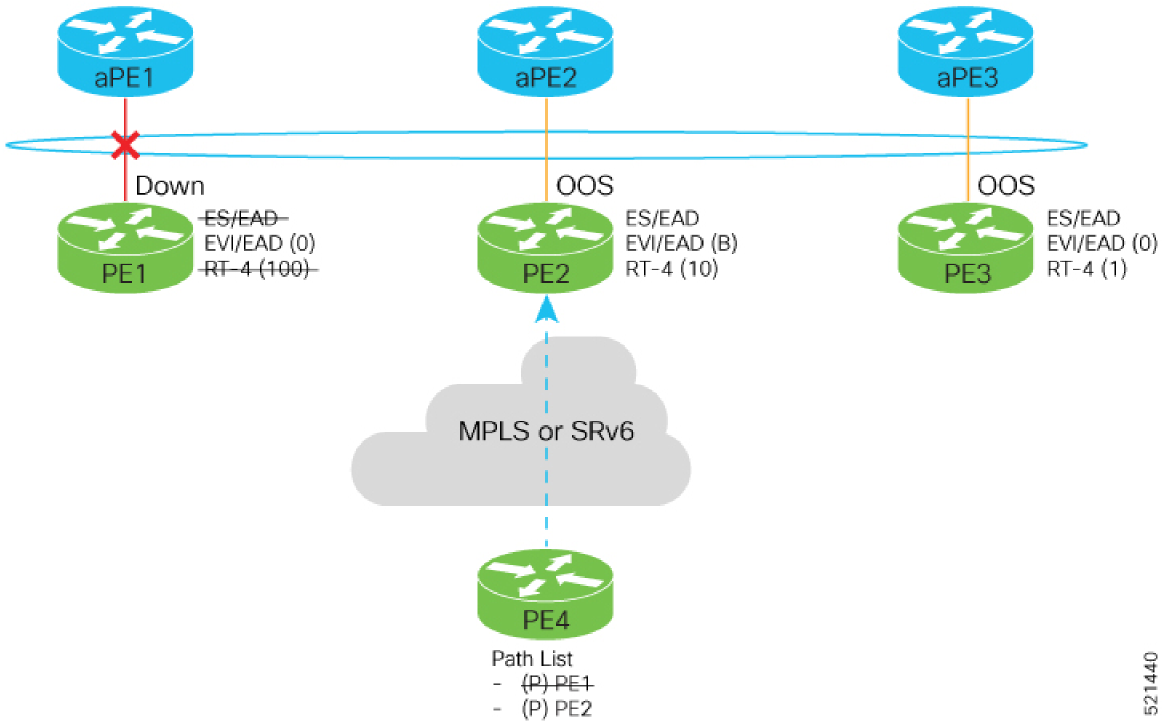
The weights configured on the EVPN PE devices cascade in the same order as the protection mechanism on the access side PEs:

- During the network failure, the redundancy ordering for the access PEs is aPE1, aPE2, aPE3.
- The weights of PE1 through PE3 are weight of PE1 > weight of PE2 > weight of PE3.

- If this ordering is not satisfied, the network will eventually converge, but it will not be as efficient as if the weights are ordered correctly.

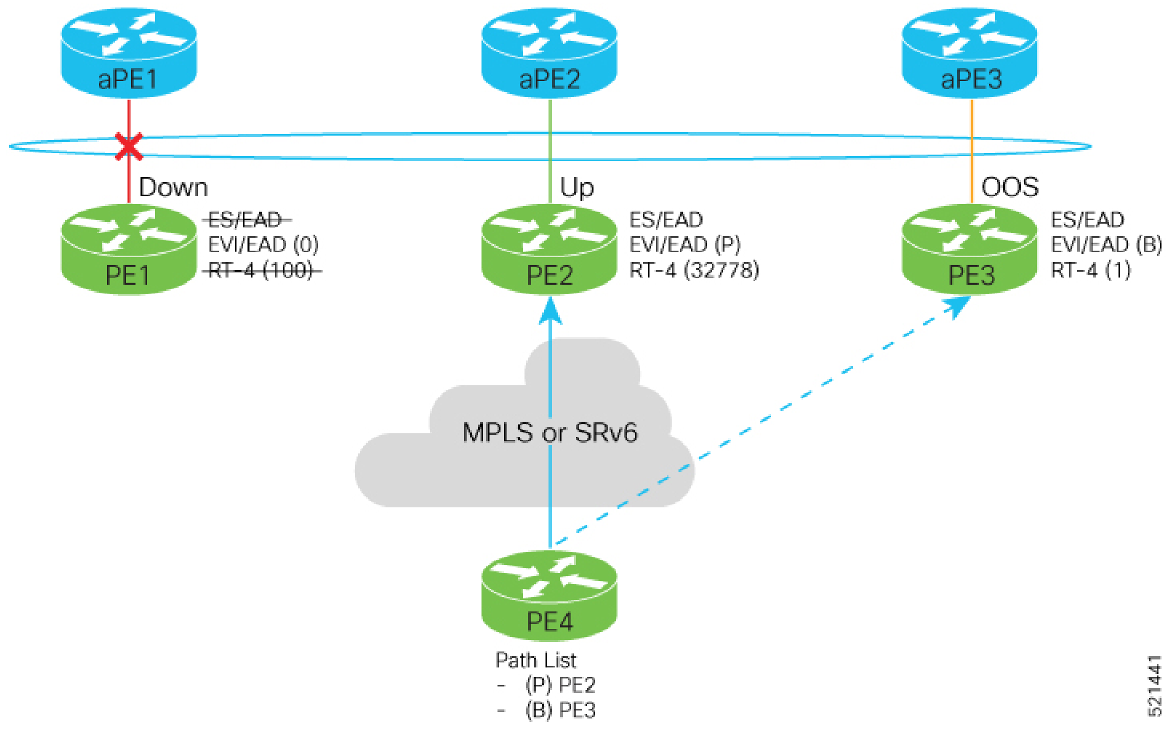
Scenario - 1

Consider a scenario where the aPE1-PE1 interface is down.



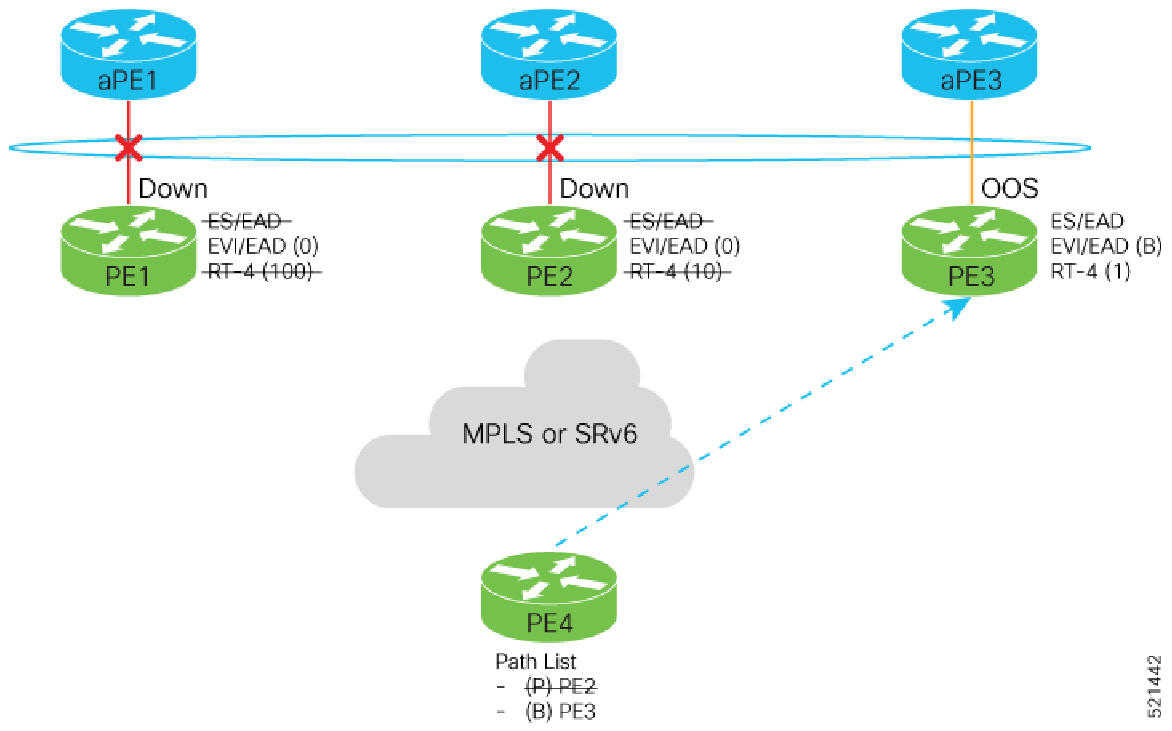
When aPE1-PE1 interface is down, the PE1 withdraws the EAD/ES route, and the traffic is sent through the backup path, which is PE2.

The aPE2-PE2 becomes the primary with a weight of 32778, and aPE3-PE3 becomes the backup. The aPE2-PE2 advertises P-bit to PE4. aPE3-PE3 advertises the B-bit to PE4.

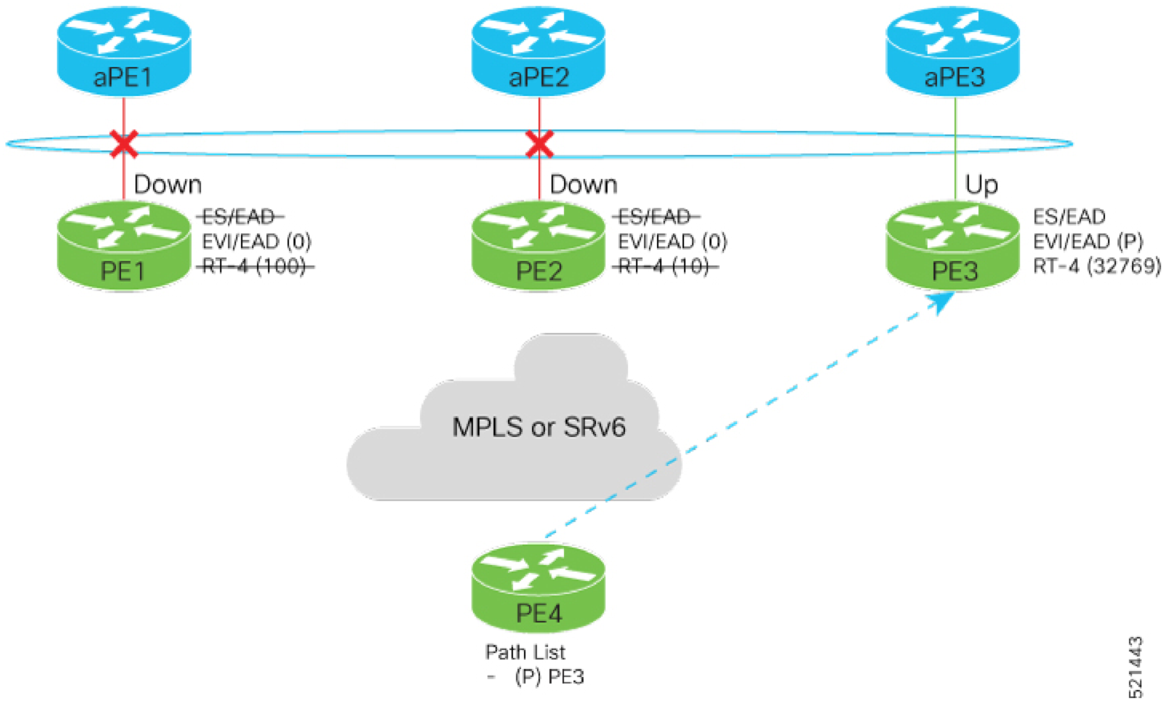


Scenario - 2

Consider a scenario where aPE2-PE2 interface is also down.

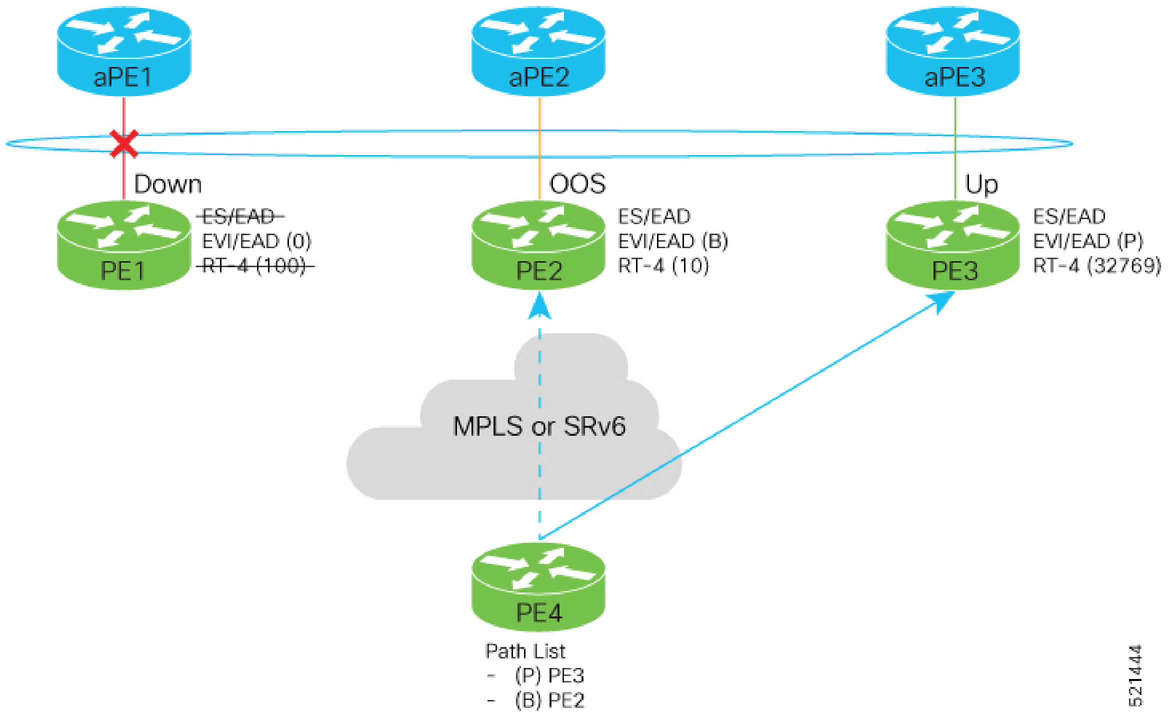


When the aPE2-PE2 interface is also down, the traffic is sent through aPE3-PE3 link. aPE3-PE3 becomes the primary path with a weight of 32769.



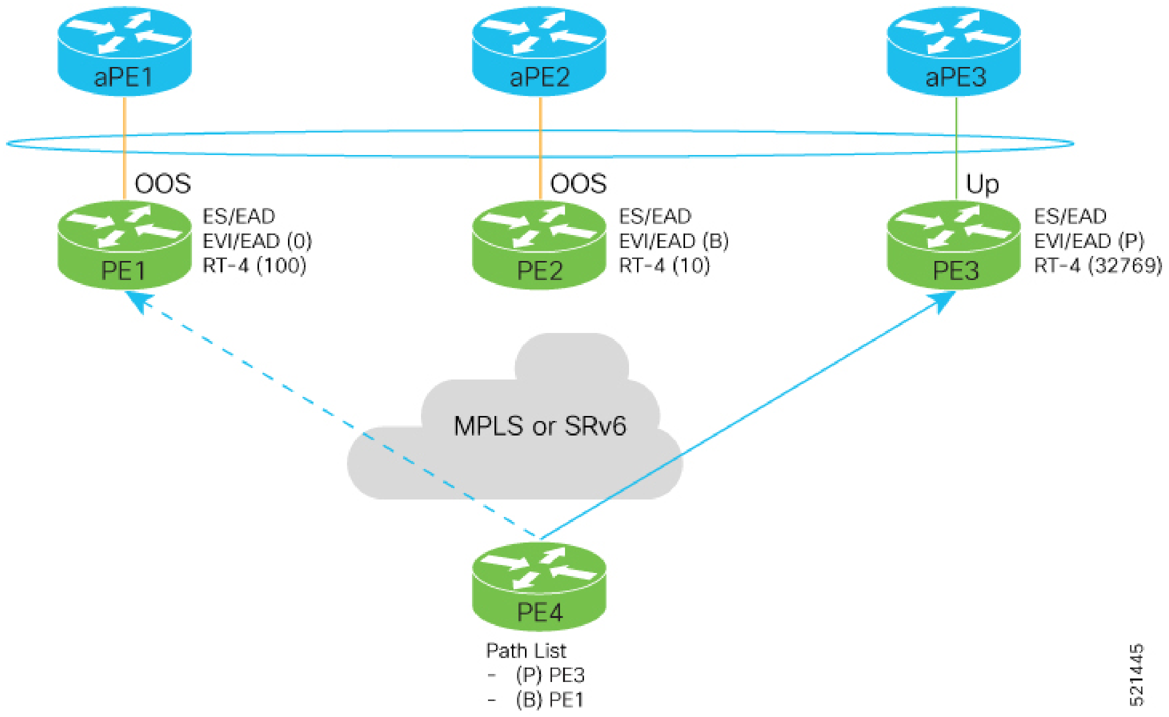
Scenario - 3

When the aPE2-PE2 interface comes up, the aPE3-PE3 link still remains the primary path. aPE2-PE2 interface becomes the backup path with a weight of 10.



Scenario - 4

When the aPE1-PE1 interface comes up, the aPE3-PE3 link remains the primary path with a weight of 32769. aPE1-PE1 interface becomes the backup path with a weight of 100. The aPE2-PE2 interface becomes NDF with a weight of 10.



Configure EVPN Access-Driven DF Election

Perform the following tasks to configure EVPN Access-Driven DF Election feature:

- Configure EVPN access-driven DF election on PE1, PE2, and PE3
- Configure LACP on aPE1, aPE2, and aPE3
- Configure EVPN-VPWS for PE1, PE2, and PE3

  See the *EVPN Virtual Private Wire Service (VPWS)* chapter on how to configure EVPN-VPWS.

Configuration Example

- All PE devices are configured with different weights. PE1, PE2, and PE3 are configured with a weight of 100, 10, and 1 respectively.
- The bundle attached to the ethernet segment is configured with **lacp mode active**.
- EVPN VPWS is configured on the PE devices.

```
/* Configure EVPN access-driven DF election on PE1, PE2, and PE3 */

/*  PE1 Configuration */
Router#configure
```

```
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 100
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit

/* PE2 Configuration */
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 10
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit

/* PE3 Configuration */
Router#configure
Router(config)#evpn
Router(config-evpn)#interface Bundle-Ether1
Router(config-evpn-ac)#ethernet-segment
Router(config-evpn-ac-es)#identifier type 0 01.11.00.00.00.00.00.00.01
Router(config-evpn-ac-es)#load-balancing-mode port-active
Router(config-evpn-ac-es)#service-carving preference-based
Router(config-evpn-ac-es-sc-pref)#weight 1
Router(config-evpn-ac-es-sc-pref)#access-driven
Router(config-evpn-ac-es-sc-pref)#commit
```

Configure LACP on aPE1, aPE2, and aPE3

```
/* aPE1 Configuration */
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/40
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 10000
Router(config-if)description Connection to PE1
Router(config-if)commit

/* aPE2 Configuration */
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/39
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 20000
Router(config-if)description Connection to PE2
Router(config-if)commit

/* aPE3 Configuration */
```

```
Router#configure
Router(config)#interface Bundle-Ether 1
Router(config-if)#lacp non-revertive
Router(config-if)#bundle maximum-active links 1 hot-standby
Router(config-if)#exit
Router(config-if)#interface GigabitEthernet0/0/0/38
Router(config-if)bundle id 10 mode active
Router(config-if)bundle port-priority 30000
Router(config-if)description Connection to PE3
Router(config-if)commit
```

Running Configuration

This section shows the running configuration of EVPN Access-Driven DF Election feature.

```
/* PE1 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 100
    access-driven
   !
 !

/* PE2 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 10
    access-driven
   !
 !

/* PE3 Configuration */
evpn
 interface Bundle-Ether 1
  ethernet-segment
   identifier type 0 01.11.00.00.00.00.00.00.01
    load-balancing-mode port-active
    service-carving preference-based
    weight 1
    access-driven
   !
 !

/* aPE1 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/40
 bundle id 10 mode active
 bundle port-priority 10000
 description Connection to PE1
!

/* aPE2 Configuration */
```

```
interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/39
 bundle id 10 mode active
 bundle port-priority 20000
 description Connection to PE2
!

/* aPE3 Configuration */

interface Bundle-Ether 1
 lacp non-revertive
  bundle maximum-active links 1 hot-standby
interface GigabitEthernet0/0/0/40
 bundle id 10 mode active
 bundle port-priority 30000
 description Connection to PE3
!
```

Verification

Verify that you have configured the EVPN Access-Driven DF Election feature successfully.

```
Router#show evpn ethernet-segment detail
Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0001.0001.0001.1b01.001b BE1                                192.168.0.1
                                                            192.168.0.3
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready
  Main port         :
     Interface name : Bundle-Ether1
     Interface MAC  : 02ef.af8d.8008
     IfHandle       : 0x00004190
     State          : Up
     Redundancy     : Active
  ESI type          : 0
     Value          : 01.0001.0001.1b01.001b
  ES Import RT      : 0100.0100.011b (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH
     Configured     : Port-Active
  Service Carving   : Preferential
     Multicast      : Disabled
  Convergence       :
  Peering Details   : 2 Nexthops
     192.168.0.1 [PREF:P:d6ce:T] >> Weight in hexadecimal
     192.168.0.3 [PREF:P:457]
  Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
  Service Carving Results:
     Forwarders     : 24
     Elected        : 6
     Not Elected    : 0
  EVPN-VPWS Service Carving Results:
     Primary        : 18
     Backup         : 0
     Non-DF         : 0
```

```
MAC Flushing mode : STP-TCN
Peering timer     : 3 sec [not running]
Recovery timer    : 30 sec [not running]
Carving timer     : 0 sec [not running]
Local SHG label   : 28384
Remote SHG labels : 0
Access signal mode: Bundle OOS (Default)
```

Associated Commands

- service-carving
- show evpn ethernet-segment

Hierarchical EVPN Access Pseudowire

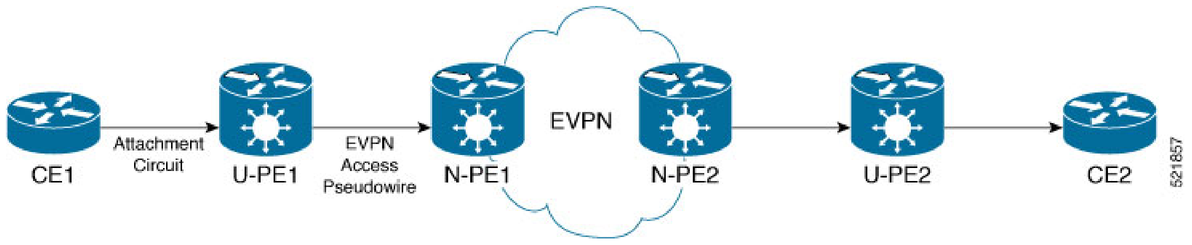
Table 21: Feature History Table

Feature Name	Release Information	Feature Description
Hierarchical EVPN Access Pseudowire	Release 7.4.1	This feature enables you to configure EVPN VPWS in the access node under the same bridge domain as EVPN in the core and helps to build a PW to the nearest high-end PE that stitches those access circuits using EVPN. Therefore, the access nodes can leverage the benefits of EVPN. This feature also allows you to reduce the number of pseudowires (PWs) between the network provider edge (N-PE) devices by replacing PE devices with user provider edge (U-PE) and network provider edge (N-PE) devices. This feature prevents signaling overhead and packet replication.

A standard VPN configuration comprises of CE devices and PE devices. With this feature, each PE device is replaced with a user provider edge (U-PE) and network provider edge (N-PE) devices. U-PE devices communicate with the CE devices and N-PE devices on the access side, and N-PE devices communicate with other N-PE devices on the core.

The Hierarchical EVPN Access Pseudowire feature allows you to reduce the number of pseudowires (PWs) between the network provider edge (N-PE) devices. The user provider edge (U-PE) device connects to the N-PE device using EVPN access pseudowire (PW) for each VPN instance. Each CE device is connected to a U-PE device through an attachment circuit.

Hierarchical EVPN Access Pseudowire Topology



In this topology, a user provider edge (U-PE1) device is connected to the CE1 through an attachment circuit. The U-PE1 device transports the CE1 traffic over an EVPN access PW to a network provider edge (N-PE1) device. The N-PE1 is connected with other N-PE2 in an EVPN core. On the N-PE1, the access PW coming from the U-PE1 is much like an AC. The U-PE is not part of the core with the other N-PEs. The N-PE forwards traffic from that access PW to the core PWs that are part of the EVPN core.

Configure Hierarchical EVPN Access Pseudowire

Perform the following task to configure Hierarchical EVPN Access Pseudowire feature on U-PEs and N-PEs.

Configuration Example

```
/* Configure U-PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group XG1
Router(config-l2vpn-xc)# p2p P1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/31
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4 target 33 source 33
Router(config-l2vpn-xc-p2p-pw)# commit

/* Configure N-PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group evpn
Router(config-l2vpn-bg)# bridge-domain evpn1
Router(config-l2vpn-bg-bd)# neighbor evpn evi 4 target 33
Router(config-l2vpn-bg-bd)# evi 1
Router(config-l2vpn-bg-bd-evi)# commit
```

Running Configuration

This section shows the Hierarchical EVPN Access Pseudowire running configuration.

```
/* U-PE1 Configuration */
l2vpn
 xconnect group XG1
  p2p P1
   interface TenGigE0/0/0/31 l2transport
   neighbor evpn evi 4 target 33 source 33
   !
!
/* N-PE1 Configuration */
l2vpn
bridge group evpn
  bridge-domain evpn1
   neighbor evpn evi 4 target 33
   evi 1
   !
  !
!
!
```

Verification

Verify the EVPN state, and the list of access PWs. The following is the sample output on N-PE1:

```
Router:N-PE1# show l2vpn bridge-domain bd-name evpn1
Wed Jun 16 09:22:30.328 EDT
Legend: pp = Partially Programmed.
Bridge group: evpn, bridge-domain: evpn1, id: 1, state: up, ShgId: 0, MSTi: 0
  Aging: 300 s, MAC limit: 4000, Action: none, Notification: syslog
  Filter MAC addresses: 0
  ACs: 0 (0 up), VFIs: 0, PWs: 1 (1 up), PBBs: 0 (0 up), VNIs: 0 (0 up)
  List of EVPNs:
    EVPN, state: up
  List of ACs:
  List of Access PWs:
      EVPN 4,33,192.168.0.4, state: up, Static MAC addresses: 0
  List of VFIs:
  List of Access VFIs:
```

Inter-AS EVPN Option B

Table 22: Feature History Table

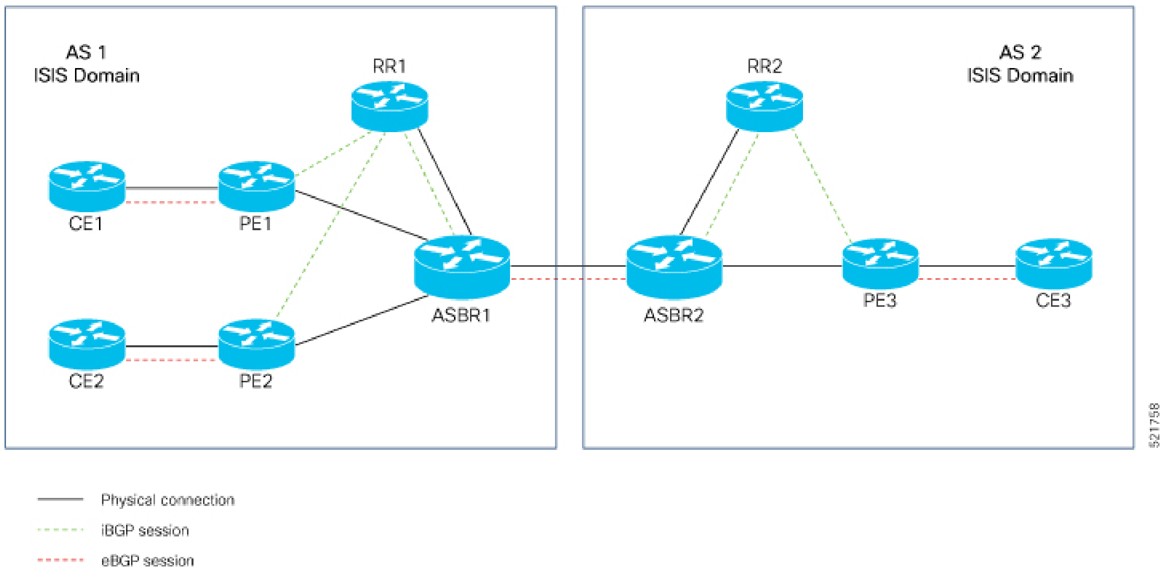
Feature Name	Release Information	Feature Description
Inter-AS EVPN Option B	Release 7.4.1	This feature enables the service providers to establish an end-to-end EVPN service over an MPLS backbone that spans multiple autonomous systems (AS). Inter-AS EVPN Option B allows the autonomous system boundary routers (ASBRs) to exchange L2VPN EVPN label routes between AS without the need for dedicated interfaces. This feature helps you to increase the number of services terminated on PE devices without requiring a dedicated number of interfaces on ASBR nodes. This feature introduces the **option-b-asbr-only** command.

The Inter-AS Option B for EVPN feature allows the service providers to offer the L2VPN EVPN service across service provider boundaries similar to L3VPN. Typically, service providers are in charge of AS and offers L2VPN EVPN services to its customers. SP customers control access devices and would want pure L2 or a combination of L2 and L3 unicast or multicast services with single or dual-homing capabilities. This is achieved by setting up MPLS tunnels over the SP core similar to L3VPN.

Prior to this release, L2VPN EVPN routes could not be exchanged across AS boundaries because ASBRs do not assign a local label to L2VPN EVPN routes. Hence L2VPN EVPN routes were not advertised to other ASBRs.

Inter-AS EVPN Option B allows L2VPN EVPN routes to be exchanged across AS boundaries because the ASBRs allocate the local label for L2VPN EVPN route types, and also perform the rewrite action. To provide an end-to-end L2VPN EVPN service across AS boundaries, you must combine the EVPN Label Switched Path (LSP) together, from PE1 to ASBR1, ASBR1 to ASBR2, and from ASBR2 to PE3.

Figure 96: Inter-AS EVPN Option B



In this topology:

- The L2VPN EVPN session between ASBRs is used to exchange the L2VPN EVPN prefixes. BGP session is used to exchange L2VPN EVPN routes between PEs and ASBRs and between ASBRs.
- A labeled switched path must exist between the PEs or each carrier. Exchange of labels is accomplished using BGP on the Inter-AS link.
- These are the three LSPs where next-hop changes:
    - PE1 to ASBR1
    - ASBR1 to ASBR2
    - ASBR2 to the PE3
- End-to-end LSPs using three hops make QoS easier to manage.
- The ASBRs are configured to change the next-hop when sending L2VPN EVPN NLRIs to the eBGP neighbors. Therefore, the ASBRs must allocate a new label when they forward the NLRI to the eBGP neighbors.
- ASBR assigns a local label to L2VPN EVPN routes and L2VPN EVPN routes are advertised to other ASBR.
- ASBRs must have all of the L2VPN EVPN prefixes, which requires them to be as resource intensive as route reflectors.

Restrictions

- Support EVPN Type-1, Type-2 (MAC only, MAC-IP with only MAC label), Type-3, and Type-5 routes.
- Type-2 MAC-IP routes with two labels, MAC label, and IP label are not supported.
- This feature does not support dual-home mode.

Configure Inter-AS EVPN Option B

Perform the following tasks to configure Inter-AS EVPN Option B:

- Configure EVPN-VPWS
- Configure native EVPN
- Configure EVPN IRB
- Configure BGP

Configuration Example

Configure EVPN-VPWS on PE1.

```
/* Type-1 Route */
Router# configure
Router(config)# interface TenGigE0/0/0/9.33 l2transport
Router(config-subif)# encapsulation dot1q 33
Router(config-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xconnect-group
Router(config-l2vpn-xc)# p2p p2p_33
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/9.33
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 4033 target 333 >> Exchange target and source
 on remote PE
Router(config-l2vpn-xc-p2p-pw)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# evi 4033
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target 4033:4033
Router(config-evpn-instance-bgp)# commit
```

Configure native EVPN on PE1.

```
/* Type-2 MAC only Route */
Router# configure
Router(config)# interface TenGigE0/0/0/9.22 l2transport
Router(config-subif)# encapsulation dot1q 22
Router(config-subif)# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group evpn-group
Router(config-l2vpn-bg)# bridge-domain evpn_3022
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/9.22
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# evi 3022
Router(config-l2vpn-bg-bd-evi)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# exit
Router(config)# exit
Router(config)# evpn
Router(config-evpn)# evi 3022
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target 3022:3022
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# advertise-ma >> advertise mac to other PEs using EVPN type-2
```

```
 routes
Router(config-evpn-instance-mac)# commit
```

Configure EVPN IRB on PE1.

```
/* Type-2 MAC-IP Route with only MAC Layer Label */
Router# configure
Router(config)# interface TenGigE0/0/0/9.12 l2transport
Router(config-subif)# encapsulation dot1q 12
Router(config-subif)# rewrite ingress tag pop 1 symmetric
Router(config-subif)# exit
Router(config)# interface BVI12 > BVI under default vrf generate type-2  mac-ip route with
 only MAC layer label
Router(config-if)# host-routing
Router(config-if)# ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)# ipv6 address 2020:c::1/112
Router(config-if)# mac-address 20.12.1
Router(config-if)# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group evpn-irb-group
Router(config-l2vpn-bg)# bridge-domain evpn_2012
Router(config-l2vpn-bg-bd)# interface TenGigE0/0/0/9.12
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface BVI12
Router(config-l2vpn-bg-bd-bvi)# split-horizon group core
Router(config-l2vpn-bg-bd-bvi)# exit
Router(config-l2vpn-bg-bd-bvi)# evi 2012
Router(config-l2vpn-bg-bd-evi)# exit
Router(config-l2vpn-bg-bd)# exit
Router(config-l2vpn-bg)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# evi 2012
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target 2012:2012
Router(config-evpn-instance-bgp)# commit
```

Configure BGP on PE1.

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# bgp router-id 10.0.0.2
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy set_community out
Router(config-bgp-nbr-af)# advertise vpnv4 unicast
Router(config-bgp-nbr-af)# advertise vpnv6 unicast >> advertise IP prefixes as type-5 routes
 under l2vpn evpn address family
Router(config-bgp-nbr-af)# vrf cust-1
Router(config-bgp-vrf)# rd 1:1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# commit
```

Configure BGP on ASRBR.

```
Router# configure
Router(config)# router bgp 1
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# label mode per-nexthop-received-label
Router(config-bgp-af)# option-b-asbr-only > Enables Inter-AS EVPN option B
Router(config-bgp-af)# retain route-target all
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 192.0.2.1
Router(config-bgp-nbr)# remote-as 2
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# neighbor 172.16.0.1
Router(config-bgp-nbr)# remote-as 1
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# next-hop-self
Router(config-bgp-nbr-af)# commit
```

Runninng Configuration

This section shows the Inter-AS EVPN Option B running configuration.

```
/* EVPN-VPWS Configuration on PE1 */
interface TenGigE0/0/0/9.33 l2transport
 encapsulation dot1q 33

l2vpn
 xconnect group xconnect-group
 p2p p2p_33
   interface TenGigE0/0/0/9.33
   neighbor evpn evi 4033 target 333 source 133

evpn
 evi 4033
  bgp
   route-target 4033:4033
!
/* Native EVPN Configuration */
interface TenGigE0/0/0/9.22 l2transport
 encapsulation dot1q 22

l2vpn
 bridge group evpn-group
  bridge-domain evpn_3022
   interface TenGigE0/0/0/9.22
   !
   evi 3022

evpn
 evi 3022
  bgp
   route-target 3022:3022
  !
  advertise-mac
!
/* EVPN IRB Configuration on PE1 */
```

```
interface TenGigE0/0/0/9.12 l2transport
 encapsulation dot1q 12
 rewrite ingress tag pop 1 symmetric

interface BVI12
 host-routing
 ipv4 address 10.0.0.1 255.0.0.0
 ipv6 address 2020:c::1/112
 mac-address 20.12.1

l2vpn
 bridge group evpn-irb-group
  bridge-domain evpn_2012
   interface TenGigE0/0/0/9.12
   !
   routed interface BVI12
    split-horizon group core
   !
   evi 2012

evpn
 evi 2012
  bgp
   route-target 2012:2012
!
/* BGP Configuration on PE1 */
router bgp 1
 bgp router-id 10.0.0.2
  address-family l2vpn evpn

 neighbor 172.16.0.1
  remote-as 1
  update-source Loopback0
  address-family l2vpn evpn
   route-policy pass-all in
   route-policy set_community out
   advertise vpnv4 unicast
   advertise vpnv6 unicast
  vrf cust-1
  rd 1:1
  address-family ipv4 unicast
   label mode per-vrf
  !
  address-family ipv6 unicast
   label mode per-vrf
  !
!
/* BGP Configuration on ASBR */
router bgp 1
 address-family l2vpn evpn
  label mode per-nexthop-received-label
  option-b-asbr-only
  retain route-target all

 neighbor 192.0.2.1
  remote-as 2
  address-family l2vpn evpn
   route-policy pass-all in
   route-policy pass-all out

 neighbor 172.16.0.1
  remote-as 1
  update-source Loopback0
  address-family l2vpn evpn
```

```
   route-policy pass-all in
   route-policy pass-all out
   next-hop-self
```

Verification

Verify the Inter-AS EVPN Option B configuration.

```
Router:PE1# show bgp l2vpn evpn rd 10.0.0.2:4033
[1][0000.0000.0000.0000.0000][133]/120 > Type - 1 route
Last Modified: Feb  3 23:05:09.595 for 00:02:35
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    172.16.0.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    172.16.0.1
  Local
    0.0.0.0 from 0.0.0.0 (10.0.0.2)
      Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
 rib-install
      Received Path ID 0, Local Path ID 1, version 153095
      Extended community: EVPN L2 ATTRS:0x06:1504 RT:4033:4033

Router:PE1# show bgp l2vpn evpn rd 10.0.0.2:3022
[2][0][48][0011.0100.00c9][0]/104
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    172.16.0.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    172.16.0.1
  Local
    0.0.0.0 from 0.0.0.0 (10.0.0.2)
      Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
 rib-install
      Received Path ID 0, Local Path ID 1, version 153097
      Extended community: SoO:10.0.0.2:3022 0x060e:0000.0000.0016 RT:3022:3022
      EVPN ESI: 0000.0000.0000.0000.0000
```

Note EVPN Option B supports Type-2 MAC-IP routes with only MAC layer labels; Type-2 MAC-IP routes with two labels, MAC layer labels, and IP layer labels are not supported.

BGP receives L2VPN EVPN routes from EVPN.

```
Router:PE1# show bgp l2vpn evpn bridge-domain evpn_2012
...
Route Distinguisher: 10.0.0.2:2012 (default for vrf evpn_2012)
*> [2][0][48][0011.0100.0065][32][20.0.12.11]/136 >> Type-2 MAC-IP routes
                      0.0.0.0                                0 i
*> [2][0][48][0011.0100.0065][128][2020:c::11]/232
                      0.0.0.0                                0 i
*> [2][0][48][0011.0100.0065][128][fe80::211:1ff:fe00:65]/232
                      0.0.0.0                                0 i
*>i[2][0][48][0012.0100.0065][32][20.0.12.51]/136
                      2.2.2.2                       100      0 I
*>i[2][0][48][0013.0100.0065][32][20.0.12.101]/136
                      3.3.3.3                       100      0 2 I
*> [3][0][32][10.0.0.2]/80 >> Type-3 Inclusive Multicast Ethernet Tag (IMET) route
```

```
                      0.0.0.0                                0 i
*>i[3][0][32][2.2.2.2]/80
                      2.2.2.2                        100      0 i
*>i[3][0][32][5.5.5.5]/80
                      3.3.3.3                        100      0 2 i

Router:PE1# show evpn evi vpn-id 2012 detail
VPN-ID     Encap      Bridge Domain                Type
---------- ---------- ---------------------------- -------------------
2012       MPLS       evpn_2012                    EVPN
   Stitching: Regular
   Unicast Label  : 26048
   Multicast Label: 24000
   ...
   BVI Subnet Withheld: ipv4 No, ipv6 No
   RD Config: none
   RD Auto  : (auto) 10.0.0.2:2012
   RT Auto  : 1:2012
   Route Targets in Use           Type
   ------------------------------ ---------------------
   2012:2012                      Both
...
```

If PE is aware of the destination MAC address, the PE uses unicast label for forwarding traffic, and if PE is not aware of the destination MAC route, multicast label is used for forwarding traffic.

Verify the ASBR BGP configuration.

```
/* Route Type-2 Verification */
Router:ASBR-1# show bgp l2vpn evpn rd 10.0.0.2:2012
[2][0][48][0011.0100.0065][32][20.0.12.11]/136
...
    Local Label: 25018
Paths: (1 available, best #1)
Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    192.0.2.1
  Local
    10.0.0.2 (metric 20) from 172.16.0.1 (10.0.0.2)
      Received Label 26048
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
not-in-vrf
      Received Path ID 1, Local Path ID 1, version 6705962
      Community: internet 1:1 2:2 3:3 4:4 5:5 6:6 7:7 8:8 9:9
      Large Community: 0:0:0 1:1:1 2:2:2 3:3:3 4:4:4 5:5:5 6:6:6 7:7:7 8:8:8 9:9:9
      Extended community: Flags 0x14: SoO:10.0.0.2:2012 0x060e:0000.0000.000c RT:2012:2012

      Originator: 10.0.0.2, Cluster list: 172.16.0.1
      EVPN ESI: 0000.0000.0000.0000.0000

/* Route Type-3 Verification */
Router:ASBR-1# show bgp l2vpn evpn rd 10.0.0.2:2012
[3][0][32][10.0.0.2]/80
...
    Local Label: 201762
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    192.0.2.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    192.0.2.1
  Local
```

```
    10.0.0.2 (metric 20) from 172.16.0.1 (10.0.0.2)
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
not-in-vrf
      Received Path ID 1, Local Path ID 1, version 893
      Community: internet 1:1 2:2 3:3 4:4 5:5 6:6 7:7 8:8 9:9
      Large Community: 0:0:0 1:1:1 2:2:2 3:3:3 4:4:4 5:5:5 6:6:6 7:7:7 8:8:8 9:9:9
      Extended community: RT:2012:2012
      Originator: 10.0.0.2, Cluster list: 172.16.0.1
      PMSI: flags 0x00, type 6, label 24000, ID 0x01010101
```

EVPN IGMPv2 Selective Multicast

Table 23: Feature History Table

Feature Name	Release Information	Feature Description
EVPN IGMPv2 Selective Multicast	Release 7.5.1	Using this feature, you can now forward multicast traffic over the EVPN network only to the receivers in the multicast groups. This targeted and selective forwarding helps eliminate unnecessary flooding of traffic. This feature also helps in optimal forwarding and efficient bandwidth utilization.

Multicast traffic is getting forwarded to all PE devices participating in a given EVPN instance, regardless of presence of interested receivers. Without BUM suppression, BUM traffic is flooded to all PE devices. This leads to very inefficient use of inter-PE bandwidth as the volume of traffic increases. For example, if multicast is used for live video feed distribution.

This feature allows the anycast gateway routers to forward multicast traffic over EVPN network to only to the receivers in the multicast groups using selective multicast.

With this feature, leaf sends the IGMP reports as BGP EVPN Route Type 6 and centralized gateway (CGW) learns it. When traffic is received over MVPN from the external source, centralized gateway sends the traffic only to interested receiver. Centralized gateway also floods the traffic to leaf which doesn't support RT-6.

You can enable this feature using **igmp-snooping** command under evpn-evi-proxy mode.

CGW supports only MVPN session profile 14 to remote PE.

Note This feature does not support IPv6 multicast and SSM.

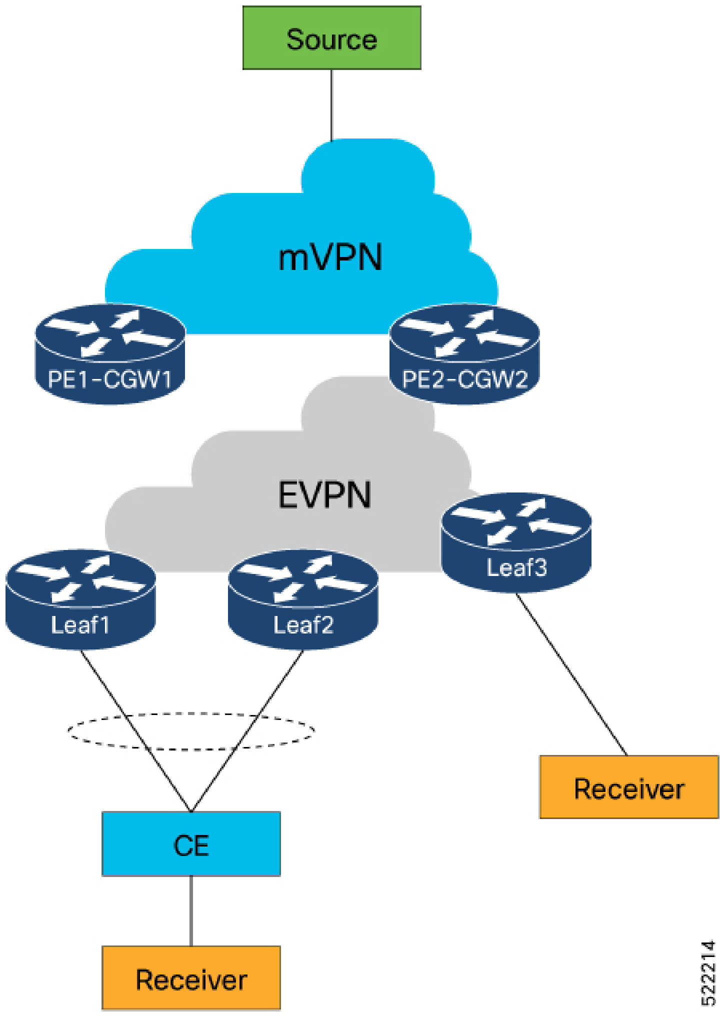
Topology

PE1-CGW1 and PE2-CGW2 are centralized gateways. Leaf 1 and Leaf 2 are multihomed. Leaf 3 is single-homed. Receivers are connected to leaf and the mulicast source is external, and are connected to centralized gateways, which is configured with mPVN profile14.

IGMP reports from the leaf are sent as BGP EVPN Route Type 6 and CGW learns it. When traffic is received over mVPN from the external source, CGW sends the traffic only to leafs which has the interested receiver.

CGW also floods the traffic to leaf which doesn't support RT-6.

Figure 97: Topology



Configuration for EVPN IGMPv2 Selective Multicast

Before your enable this feature, perform the following tasks on all nodes:

- Configure BGP
- Configure MPLS
- Enable multicast
- BVI configuration

The following is the BGP configuration:

```
router bgp 1
 nsr
 bgp router-id 1.1.1.21
 bgp graceful-restart
 address-family vpnv4 unicast
 !
 address-family vpnv6 unicast
 !
 address-family ipv4 mvpn
 !
 address-family l2vpn evpn
 !
 neighbor 1.1.1.11
  remote-as 1
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
 neighbor 1.1.1.12
  remote-as 1
  update-source Loopback0
  address-family l2vpn evpn
  !
 !
vrf cgw
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
 vrf vrf10
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
  address-family ipv4 mvpn
  !
 !
```

The following is the MPLS configuration:

```
mpls ldp
 graceful-restart
 mldp
  address-family ipv4
  !
 !
 router-id 1.1.1.21
 interface Bundle-Ether22
 !
 interface Bundle-Ether222
 !
 interface Bundle-Ether2222
 !
 interface HundredGigE0/0/0/1
 !
 interface HundredGigE0/0/0/3
 !
 interface HundredGigE0/0/0/5
 !

l2vpn
 bridge group bg
  bridge-domain cgw10
   igmp snooping profile snoop_profile1
   access-evi 10
   routed interface BVI10
   !
  !
  bridge-domain cgw11
   igmp snooping profile snoop_profile1
   access-evi 11
   routed interface BVI11
   !
```

The following is the multicast configuration:

```
multicast-routing
 address-family ipv4
  mdt source Loopback0
  interface all enable
  bgp auto-discovery mldp
  !
 vrf cgw
  address-family ipv4
   mdt source Loopback0
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery mldp
   !
   mdt partitioned mldp ipv4 p2mp
   mdt data mldp 1000
  !
 !
 vrf vrf10
  address-family ipv4
   mdt source Loopback0
   rate-per-route
   interface all enable
   accounting per-prefix
   bgp auto-discovery mldp
   !
   mdt partitioned mldp ipv4 p2mp
   mdt data mldp 1000
 !

router pim
 vrf cgw
  address-family ipv4
   rpf topology route-policy rpf-profile14
   mdt c-multicast-routing bgp
   !
   rp-address 1.1.1.113
  !
 !
 vrf vrf10
  address-family ipv4
   rpf topology route-policy rpf-profile14
   mdt c-multicast-routing bgp
```

```
    !
    rp-address 1.1.1.114
   !
  !
  vrf vrf11
   address-family ipv4
    rpf topology route-policy rpf-profile14
    mdt c-multicast-routing bgp
    !
    rp-address 1.1.1.115
   !
  !
vrf vrf12
   address-family ipv4
    rpf topology route-policy rpf-profile14
    mdt c-multicast-routing bgp
    !
    rp-address 1.1.1.116
   !
  !
  vrf vrf13
   address-family ipv4
    rpf topology route-policy rpf-profile14
    mdt c-multicast-routing bgp
    !
    rp-address 1.1.1.117
   !
  !
  vrf vrf14
   address-family ipv4
    rpf topology route-policy rpf-profile14
    mdt c-multicast-routing bgp
    !
    rp-address 1.1.1.118
   !
  !
```

The following is the BVI configuration:

```
interface BVI10
 vrf cgw
 ipv4 address 10.10.1.1 255.255.0.0
 ipv6 address 10:10::1/64
 mac-address 11.d1.d2
!
interface BVI11
 vrf cgw
 ipv4 address 10.11.1.1 255.255.0.0
 ipv6 address 10:11::1/64
 mac-address 0.d3.d4
!
```

Configuration for Centralized Gateway

L2VPN and EVPN configuration:

The following is the configuration of centralized gateway routers:

```
Router#configure
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg
Router(config-l2vpn-bg)#bridge-domain cgw10
Router(config-l2vpn-bg-bd)#igmp snooping profile snoop_profile1
Router(config-l2vpn-bg-bd)#access-evi 10
Router(config-l2vpn-bg-bd)#routed interface BVI10
Router(config-l2vpn-bg-bd-bvi)#exit
Router(config-l2vpn-bg-bd)#exit

Router(config-l2vpn-bg)#bridge-domain cgw11
Router(config-l2vpn-bg-bd)#igmp snooping profile snoop_profile1
Router(config-l2vpn-bg-bd)#access-evi 11
Router(config-l2vpn-bg-bd)#routed interface BVI1
Router(config-l2vpn-bg-bd-bvi)#


Router(config)#evpn
Router(config-evpn)# evi 10
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#bvi-mac
Router(config-evpn-instance-mac)#proxy
Router(config-evpn-instance-proxy)# igmp-snooping
Router(config-evpn-instance-proxy)#exit
Router(config-evpn-instance)#exit

Router(config-evpn)#evi 11
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#proxy
Router(config-evpn-instance-proxy)#igmp-snooping
Router(config-evpn-instance-proxy)#
```

IGMP snooping configuration:

```
Router(config)#igmp snooping profile snoop_profile1
Router(config-igmp-snooping-profile)#
```

Configuration for Leafs

The following is the configuration of Leaf routers

```
Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg
Router(config-l2vpn-bg)#bridge-domain cgw10
Router(config-l2vpn-bg-bd)#multicast-source ipv4
Router(config-l2vpn-bg-bd)#igmp snooping profile snoop_profile1
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/0/0/0.10
Router(config-l2vpn-bg-bd-ac)#exit
Router(config-l2vpn-bg-bd)#exit

Router(config-l2vpn)#evpn
Router(config-evpn)#evi 10
Router(config-evpn-instance)#advertise-mac
Router(config-evpn-instance-mac)#exit
Router(config-evpn-instance-proxy)#igmp-snooping
```

The following is the IGMP snooping configuration for leaf router:

```
Router(config)#igmp snooping profile snoop_profile1
Router(config-igmp-snooping-profile)#system-ip-address 1.1.1.11
Router(config-igmp-snooping-profile)#internal-querier
Router(config-igmp-snooping-profile)#internal-querier version 3
Router(config-igmp-snooping-profile)#internal-querier query-interval 60
```

The followung the route-policy configuration:

```
route-policy policy-discard-smet-routes
  if evpn-route-type is 6 then
    drop
  else
    pass
  endif
end-policy
!
```

Verification

```
Router<PE1-CGW1># show igmp snooping port group 226.43.1.3

Key: GM=Group Filter Mode, PM=Port Filter Mode
Flags Key: S=Static, D=Dynamic, B=BGP Learnt, E=Explicit Tracking, R=Replicated

                            Bridge Domain group1:cgw830

Port                        PM Group           Ver GM Source          Exp   Flgs
----                        -- -----           --- -- ------          ---   ----
Ev830,Peer 1.1.1.11         -  226.43.1.3      V2  -  *               never B

                            Bridge Domain group1:cgw831

Port                        PM Group           Ver GM Source          Exp   Flgs
----                        -- -----           --- -- ------          ---   ----
Ev831,Peer 1.1.1.11         -  226.43.1.3      V2  -  *               never B

                             Bridge Domain group1:cgw837

Port                        PM Group           Ver GM Source          Exp   Flgs
----                        -- -----           --- -- ------          ---   ----
Ev837,Peer 1.1.1.11         -  226.43.1.3      V2  -  *               never B

                            Bridge Domain group1:cgw838

Port                        PM Group           Ver GM Source          Exp   Flgs
----                        -- -----           --- -- ------          ---   ----
Ev838,Peer 1.1.1.11         -  226.43.1.3      V2  -  *               never B

                            Bridge Domain group1:cgw839

Port                        PM Group           Ver GM Source          Exp   Flgs
----                        -- -----           --- -- ------          ---   ----
Ev839,Peer 1.1.1.11         -  226.43.1.3      V2  -  *               never B

Router<PE2-CGW2># show evpn ethernet-segment carving detail
Legend:
  B   - No Forwarders EVPN-enabled,
  C   - MAC missing (Backbone S-MAC PBB-EVPN / Grouping ES-MAC vES)
  RT  - ES-Import Route Target missing,
  E   - ESI missing,
  H   - Interface handle missing,
  I   - Name (Interface or Virtual Access) missing,
  M   - Interface in Down state,
  O   - BGP End of Download missing,
  P   - Interface already Access Protected,
  Pf  - Interface forced single-homed,
  R   - BGP RID not received,
  S   - Interface in redundancy standby state,
  X   - ESI-extracted MAC Conflict
  SHG - No local split-horizon-group label allocated
  Hp  - Interface blocked on peering complete during HA event
```

```
Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0000.00ac.ce55.00e1.0000 Access-EVI:all              1.1.1.21
                                                            1.1.1.22
  ES to BGP Gates   : Ready
  ES to L2FIB Gates : Ready

Main port         :
     Interface name : Access-EVI/all
     Interface MAC  : 0000.0000.0000
     IfHandle       : 0x00000000
     State          : Up
     Redundancy     : Not Defined
  ESI type          : 0
     Value          : 00.00ac.ce55.00e1.0000
  ES Import RT      : 0000.acce.5500 (from ESI)
  Source MAC        : 0000.0000.0000 (N/A)
  Topology          :
     Operational    : MH, Single-active
     Configured     : Single-active (AApS) (default)
  Service Carving   : Auto-selection
     Multicast      : Disabled
  Convergence       :
  Peering Details   : 2 Nexthops
     1.1.1.21 [MOD:P:7fff:T]
     1.1.1.22 [MOD:P:00:T]
Service Carving Synchronization:
     Mode           : NONE
     Peer Updates   :
                1.1.1.21 [SCT: 2021-11-16 10:12:21.1637086]
                1.1.1.22 [SCT: N/A]
  Service Carving Results:
     Forwarders     : 1690
     Elected        : 845
            EVI E   :       10,      12,      14,      16,      18,      20
            EVI E   :       22,      24,      26,      28,      30,      32,
            EVI E   :       34,      36,      38,      40,      42,      44,
            EVI E   :       46,      48,      50,      52,      54,      56,
            EVI E   :       58,      60,      62,      64,      66,      68,
            EVI E   :       70,      72,      74,      76,      78,      80,
            EVI E   :       82,      84,      86,      88,      90,      92,
            EVI E   :       94,      96,      98,     100,     102,     104,
            EVI E   :      106,     108,     110,     112,     114,     116,
            EVI E   :     1894,    1896,    1898,    1900,    1902,    1904,
            EVI E   :     1906,    1908,    1910,    1912,    1914,    1916,
            EVI E   :     1918,    1920,    1922,    1924,    1926,    1928,
            EVI E   :     1930,    1932,    1934,    1936,    1938,    1940,
            EVI E   :     1942,    1944,    1946,    1948,    1950,    1952,
            EVI E   :     1954,    1956,    1958,    1960,    1962,    1964,
            EVI E   :     1966,    1968,    1970,    1972,    1974,    1976,
            EVI E   :     1978,    1980,    1982,    1984,    1986,    1988,
            EVI E   :     1990,    1992,    1994,    1996,    1998
Not Elected    : 845
            EVI NE  :       11,      13,      15,      17,      19,      21
            EVI NE  :       23,      25,      27,      29,      31,      33,
            EVI NE  :       35,      37,      39,      41,      43,      45,
            EVI NE  :       47,      49,      51,      53,      55,      57,
            EVI NE  :       59,      61,      63,      65,      67,      69,
            EVI NE  :       71,      73,      75,      77,      79,      81,
            EVI NE  :       83,      85,      87,      89,      91,      93,
            EVI NE  :       95,      97,      99,     101,     103,     105...
            EVI NE  :     1859,    1861,    1863,    1865,    1867,    1869,
            EVI NE  :     1871,    1873,    1875,    1877,    1879,    1881,
            EVI NE  :     1883,    1885,    1887,    1889,    1891,    1893,
            EVI NE  :     1895,    1897,    1899,    1901,    1903,    1905,
            EVI NE  :     1907,    1909,    1911,    1913,    1915,    1917,
            EVI NE  :     1919,    1921,    1923,    1925,    1927,    1929,
            EVI NE  :     1931,    1933,    1935,    1937,    1939,    1941,
            EVI NE  :     1943,    1945,    1947,    1949,    1951,    1953,
            EVI NE  :     1955,    1957,    1959,    1961,    1963,    1965,
            EVI NE  :     1967,    1969,    1971,    1973,    1975,    1977,
            EVI NE  :     1979,    1981,    1983,    1985,    1987,    1989,
            EVI NE  :     1991,    1993,    1995,    1997,    1999
EVPN-VPWS Service Carving Results:
     Primary        : 0
     Backup         : 0
     Non-DF         : 0
  MAC Flushing mode : STP-TCN
  Peering timer     : 3 sec [not running]
  Recovery timer    : 30 sec [not running]
  Carving timer     : 0 sec [not running]
  HRW Reset timer   : 5 sec [not running]
  Local SHG label   : 27884
  Remote SHG labels : 0
  Access signal mode: Unsupported
```

CHAPTER 12

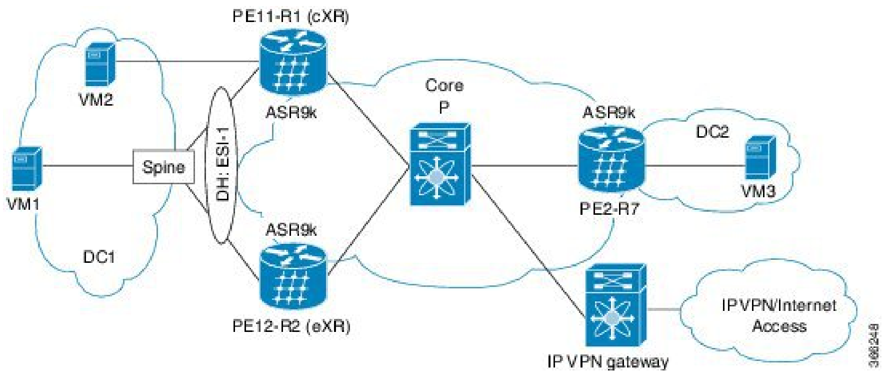
Configure EVPN IRB

This chapter introduces you to Ethernet VPN (EVPN) Integrated Routing and Bridging (IRB) feature and describe how you can configure the EVPN IRB feature.

EVPN IRB

Ethernet VPN (EVPN) provides an extensible and flexible multi-homing VPN solution for Layer 2 connectivity among hosts over an MPLS core/IP network. EVPN Integrated Routing and Bridging (IRB) feature enables Layer 3 forwarding among hosts across different IP subnets, while maintaining the multi-homing capabilities of EVPN. Also, EVPN IRB feature enables EVPN hosts or subnets to communicate with IP VPNs.

Figure 98: EVPN IRB



The above figure illustrates a scenario where EVPN IRB is deployed using three EVPN PE routers that provide single-homing or multi-homing access. The PE routers exchange EVPN and IP VPN BGP address-families information. In Cisco ASR 9000 Series Aggregation Services Router, the centralized EVPN gateway with both EVPN and VPN address-families are enabled, so EVPN route stitching and re-origination does not occur. The single IP VPN gateway provides access to other IP VPNs or the Internet. The IP VPN gateway exchanges VPNv4 or VPNv6 address families with the EVPN PE routers but EVPN address family is not available.

In the above figure, the host sends an IP packet whose frame contain its MAC and IP address. The frame of the packet also includes the destination MAC address of the BVI interface of the local PE device and the IP address of the destination host. When the host is required to send packets for routing to other subnets and IP VPN or the Internet, the host uses the MAC address of the local BVI interface as destination MAC address. The IRB interface receives the packet in whose frame the interface's MAC address is set as the destination. The IRB interface routes the packet to the destination after looking up the VRF table. The PE device chooses an intra-subnet route registered earlier in the VRF table. The MAC address that matches the destination IP, in the earlier route, becomes the new MAC address which replaces the existing MAC address in the packet. The packet reaches the destination remote PE device, to which the destination host is connected, which does an inter-subnet looks up in the VRF table for the MAC address of the destination host. Then, through the IRB interface, the remote PE device forwards the packet to the destination host.

EVPN Single-Homing Access EVPN Gateway

The EVPN provider edge (PE) devices learn the MAC address and IP address from the ARP traffic that they receives from the customer edge (CE) devices. The PEs create the MAC+IP routes. The PEs advertise the MAC+IP routes to MPLS core. They inject the host IP routes to IP-VPN gateway. All the PE nodes add the host routes in the IP-VRF table. The EVPN PE nodes add MAC route to the MAC-VRF table. The IP-VPN PE advertise the subnet routes to the provider edge devices who add the subnet routes to IP-VRF table. On the PE devices, IRB IP addresses and MAC addresses are not advertised through BGP. The IRB IP addresses or MAC addresses are used to send ARP requests towards the datacenter CEs.

EVPN Multi-Homing Active-Active

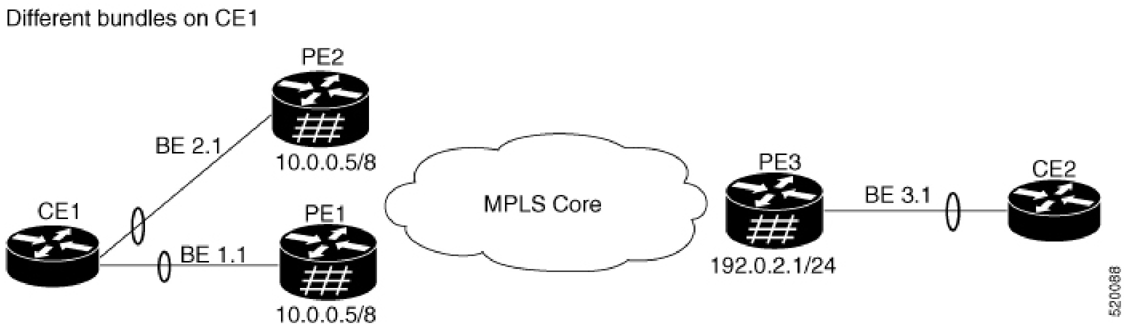
EVPN Multi-homing access gateway enables redundant network connectivity by allowing a CE device to connect to more than one PE devices. Disruptions to the network connectivity are prevented by allowing a CE device to be connected to a PE device or several PE devices through multi-homing. Ethernet segment is the bunch of Ethernet links through which a CE device is connected to more than one PE devices.

In EVPN IRB, both EVPN and IP VPN (both VPNv4 and VPNv6) address families are enabled between ASR 9000 Data Center Interconnect (DCI) gateways. When Layer 2 (L2) stretch is not available in multiple datacenters (DC), routing is established through VPNv4 or VPNv6 routes. When Layer 2 stretch is available, host routing is applied where IP-MAC routes are learnt by ARP/IPv6 ND and are distributed to EVPN/BGP. In remote peer gateway, these IP-MAC EVPN routes are imported into IP VPN routing table from EVPN route-type 2 routes with secondary label and Layer 3 VRF route-target.

EVPN Single-Active Multihoming for Anycast Gateway IRB

The EVPN Single-Active Multihoming for Anycast Gateway IRB feature supports single-active redundancy mode. In this mode, the provider edge (PE) nodes locally connected to an Ethernet Segment load balance traffic to and from the Ethernet Segment based on EVPN service instance (EVI). Within an EVPN service instance, only one PE forwards traffic to and from the Ethernet Segment (ES). This feature supports intersubnet scenario only.

Figure 99: EVPN: Single-Active Multihoming for Anycast Gateway IRB



Consider a topology where CE1 is multihomed to PE1 and PE2. Bundle Ethernet interfaces BE 1.1, BE 2.1, and the ingress interface must belong to the same switching domain on CE1. Enable host routing and configure anycast gateway IP address on both these peering PEs. PE1 and PE2 are connected to PE3 through MPLS core. PE3 has reachability of subnet 10.0.0.5/8 to both peering PEs. Peering PEs has reachability to PE3 subnet 192.0.2.1/24. CE2 is connected to PE3 through an Ethernet interface bundle. PE1 and PE2 advertise Type four routes, and then performs designated forwarder (DF) election. The non-DF blocks the traffic in both the directions in single-active mode.

Consider traffic flow from CE1 to CE2. CE1 sends an Address Resolution Protocol (ARP) broadcast request to both PE1 and PE2. Peering PEs performs designated forwarder (DF) election for shared ESI. If PE1 is the designated forwarder for the EVI, PE1 replies to the ARP request from CE1. PE2 drops the traffic from CE1. Thereafter, all the unicast traffic is sent through PE1. PE2 is set to standby or blocked state and traffic is not sent over this path. PE1 advertises MAC to PE3. PE3 always sends and receives traffic through PE1. PE3 sends the traffic to CE2 over Ethernet interface bundle. If BE1 fails, PE2 becomes the DF and traffic flows through PE2.

Configure EVPN Single-Active Multihoming

Perform the following tasks on PE1 and PE2 to configure EVPN Single-Active Multihoming feature:

- Configure EVPN IRB with host routing
- Configure EVPN Ethernet Segment

- Configure Layer 2 Interface
- Configure a Bridge Domain
- Configure VRF

Configure EVPN IRB with Host Routing

Perform this task to configure EVPN IRB with host routing.

Configuration Example

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 6005
Router(config-l2vpn-bg)# bridge-domain 6005
Router(config-l2vpn-bg-bd)# routed interface BVI50
Router(config-l2vpn-bg-bd-bvi)# exit
Router(config-l2vpn-bg-bd-bvi)# interface Bundle-Ether2.1
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpnbg-bd-evi)# commit
Router(config-l2vpnbg-bd-evi)# exit
Router(config)# interface BVI50
Router(config-if)# host-routing
Router(config-if)# vrf 30
Router(config-if)# ipv4 address 10.0.0.5 255.0.0.0
Router(config-if)# local-proxy-arp
Router(config-if)# mac-address 1.1.1
Router(config-if)# commit
```

Running Configuration

This section shows EVPN IRB with host routing running configuration.

```
configure
l2vpn
 bridge group 6005
  bridge-domain 6005
   interface Bundle-Ether2.1
    evi 6005
!
   !
 interface BVI34
 host-routing
 vrf 30
 ipv4 address 10.0.0.5 255.0.0.0
 arp learning local
 local-proxy-arp
 mac-address 1.1.1
```

Configure EVPN Ethernet Segment

Perform this task to configure the EVPN Ethernet segment.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 40.00.00.00.00.00.00.00.01
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# bgp route-target 4000.0000.0001
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
configure
evpn
 interface Bundle-Ether1
   ethernet-segment
    identifier type 0 40.00.00.00.00.00.00.00.01
    load-balancing-mode single-active
    bgp route-target 4000.0000.0001
    !
!
!
```

Configure EVPN Service Instance (EVI) Parameters

Perform this task to define EVPN service instance (EVI) parameters.

```
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 6005
Router(config-evpn-evi)# bgp
Router(config-evpn-evi-bgp)# rd 200:50
Router(config-evpn-evi-bgp)# route-target import 100:6005
Router(config-evpn-evi-bgp)# route-target export 100:6005
Router(config-evpn-evi-bgp)# commit
```

Running Configuration

```
configure
 evpn
  evi 6005
   bgp
    rd 200:50
    route-target import 100:6005
    route-target export 100:6005
!
!
```

Configure Layer 2 Interface

Perform this task to define Layer 2 interface.

```
Router# configure
Router(config)# interface bundle-ether2.1 l2transport
Router(config-subif-l2)# no shutdown
Router(config-subif-l2)# encapsulation dot1q 1
Router(config-subif-l2)# rewrite ingress tag pop 1 symmetric
Router(config-subif-l2)#commit
Router(config-subif-l2)#exit
```

Running Configuration

This section shows the Layer 2 interface running configuration.

```
configure
 interface bundle-ether2.1 l2transport
  no shutdown
  encapsulation dot1q 1
  rewrite ingress tag pop 1 symmetric
!
```

Configure a Bridge Domain

Perform the following steps to configure the bridge domain on PE1 and PE2.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 6005
Router(config-l2vpn-bg)# bridge-domain 6005
Router(config-l2vpn-bg-bd)# interface Bundle-Ether2.1
Router(config-l2vpn-bg-bd-ac)# evi 6005
Router(config-l2vpnbg-bd-evi)# commit
Router(config-l2vpnbg-bd-evi)# exit
```

Running Configuration

This section shows the bridge domain running configuration.

```
configure
l2vpn
 bridge group 6005
  bridge-domain 6005
   interface Bundle-Ether2.1
    evi 6005
!
```

Configure VRF

Perform this task to configure VRF.

Configuration Example

```
Router# configure
Router(config)# vrf 30
Router(config-vrf)# address-family ipv4 unicast
Router(config-l2vpn-vrf-af)# route-target import 100:6005
Router(config-l2vpn-vrf-af)# route-target export 100:6005
Router(config-l2vpn-vrf-af)# commit
```

Running Configuration

This section shows the VRF running configuration.

```
configure
 vrf 30
  address-family ipv4 unicast
   route-target import 100:6005
   route-target export 100:6005

!
```

EVPN IRB Support

EVPN IRB supports the following scenarios:

- In single-homing scenario, only physical, VLAN, .1q, .1ad, or QinQ access methods are supported.
- In dual-homing scenario, only two PE gateways in a redundancy group are supported.
- Both IPv4 and IPv6 are supported.

Distributed Anycast Gateway

EVPN IRB for the given subnet is configured on all the EVPN PEs that are hosted on this subnet. To facilitate optimal routing while supporting transparent virtual machine mobility, hosts are configured with a single default gateway address for their local subnet. That single (anycast) gateway address is configured with a single (anycast) MAC address on all EVPN PE nodes locally supporting that subnet. This process is repeated for each locally defined subnet requires Anycast Gateway support.

The host-to-host Layer 3 traffic, similar to Layer 3 VPN PE-PE forwarding, is routed on the source EVPN PE to the destination EVPN PE next-hop over an IP or MPLS tunnel, where it is routed again to the directly connected host. Such forwarding is also known as Symmetric IRB because the Layer 3 flows are routed at both the source and destination EVPN PEs.

The following solutions are part of the Distributed Anycast Gateway feature:

EVPN IRB with Active-Active Multi-Homing with Subnet Stretch or Host-Routing across the Fabric

For a bridge domain or subnet that is stretched across remote EVPN PEs, both /32 host routes and MAC routes are distributed in a EVPN overlay control plane to enable Layer 2 and Layer 3 traffic to the end points in a stretched subnet.

This type of multi-homing has the following characteristics:

- Layer 2 or Layer 3 ECMP for the fabric for dual-homed hosts based on Route Type 1 and Route Type 2
- Layer 3 unipath over the Fabric for single-homed hosts based on Route Type 2
- Layer 2 subnet stretch over the fabric
- Layer 2 stretch within redundancy group of leafs with orphan ports

MAC and IP Unicast Control Plane

This use case has following types:

Prefix Routing or No Subnet Stretch

IP reachability across the fabric is established using subnet prefix routes that are advertised using EVPN Route Type 5 with the VPN label and VRF RTs. Host ARP and MAC sync are established across multi-homing

EVPN PEs using MAC+IP Route Type 2 based on a shared ESI to enable local switching through both the multi-homing EVPN PEs.

Host Routing or Stretched Subnet

When a host is discovered through ARP, the MAC and IP Route Type 2 is advertised with both MAC VRF and IP VRF router targets, and with VPN labels for both MAC-VRF and IP-VRF. Particularly, the VRF route targets and Layer 3 VPN label are associated with Route Type 2 to achieve PE-PE IP routing identical to traditional L3VPNs. A remote EVPN PE installs IP/32 entries directly in Layer 3 VRF table through the advertising EVPN PE next-hop with the Layer 3 VPN label encapsulation, much like a Layer 3 VPN imposition PE. This approach avoids the need to install separate adjacency rewrites for each remote host in a stretched subnet. Instead, it inherits a key Layer 3 VPN scale benefit of being able to share a common forwarding rewrite or load-balance resource across all IP host entries reachable through a set of EVPN PEs.

ARP and MAC sync

For hosts that are connected through LAG to more that one EVPN PE, the local host ARP and MAC entries are learnt in data plane on either or both of the multihoming EVPN PEs. Local ARP and MAC entries are synced across the two multihoming EVPN PEs using MAC and IP Route Type 2 based on a shared ESI to enable local switching through both the multihoming EVPN PEs. Essentially, a MAC and IP Route Type 2 that is received with a local ESI causes the installation of a synced MAC entry that points to the local AC port, and a synced ARP entry that is installed on the local BVI interface.

Note Only one Ethernet Flow Point (EFP) is supported per non-Zero ESI per bridge domain or EVI. This is a limitation of EVPN.

MAC and IP Route Re-origination

MAC and IP Route Type 2 received with a local ESI, which is used to sync MAC and ARP entries, is also re-originated from the router that installs a SYNC entry, if the host is not locally learnt and advertised based on local learning. This route re-origination is required to establish overlay IP ECMP paths on remote EVPN PEs, and to minimize traffic hit on local AC link failures, that can result in MAC and IP route withdraw in the overlay.

Intra-subnet Unicast Data Plane

The Layer 2 traffic is bridged on the source EVPN PE using ECMP paths to remote EVPN PEs, established through MAC+IP RT2, for every ES and for every EVI, ES and EAD Route Type 2 routes that are advertised from the local EVPN PEs.

Inter-subnet Unicast Data Plane

Inter-subnet traffic is routed on the source EVPN PEs through overlay ECMP to the destination EVPN PEs next-hops. Data packet are encapsulated with the VPN label advertised from the EVPN PE and tunnel label for the BGP next-hop towards the spine. It is then routed again on the destination EVPN PE using a local ARP adjacency towards the host. IP ECMP on the remote EVPN PEs is established through local and re-originated routes advertised from the local EVPN PEs.

Centralized Anycast Gateway for EVPN IRB

Table 24: Feature History Table

Feature Name	Release Information	Feature Description
Centralized Anycast Gateway for EVPN IRB	Release 7.5.1	This feature enables configuring the IRB interface in a central location called the Centralized Anycast Gateway. Such a centralized, single-point makes it easier to configure and maintain, reducing the overall operational cost. It also makes it easy to configure the IRB because it's configured only on the centralized gateway of edge L2 devices. This feature is supported only on Single-active mode and this feature introduces **virtual access-evi** command.

EVPN Integrated Routing and Bridging (IRB) allows hosts across the overlay to communicate with each other across the subnets in the VPN. Distributed anycast gateway allows you to configure EVPN IRB for the given subnet on all the access side EVPN PEs that are hosted on the subnet. The distributed gateway provides inner and intra subnet optimal forwarding.

However, the distributed anycast gateway doesn’t provide a centralized point to perform services such as QOS and ACL. Moreover, distributed gateway brings the complexity of setting up and maintaining IRB L3 domains across leaf or edge nodes. Distributed anycast is a bit complex for mobile networks. Customers are looking at connecting and aggregating mobile antennas using cell site routers to core backbone networks in a simple way especially when access networks aren't complex or multilayered Cos fabrics. There’s a need for a single gateway device to carry the load at a lesser cost and that is easy to configure.

With this feature, you can add a centralized gateway to configure IRB in the central location that is known as Centralized Anycast Gateway. The gateway is centralized on a pair of nodes that act as an interface point between the layer-2 and layer-3 routed networks. The Centralized Anycast Gateway uses anycast IRB.

Centralized Anycast GW supports the following services:

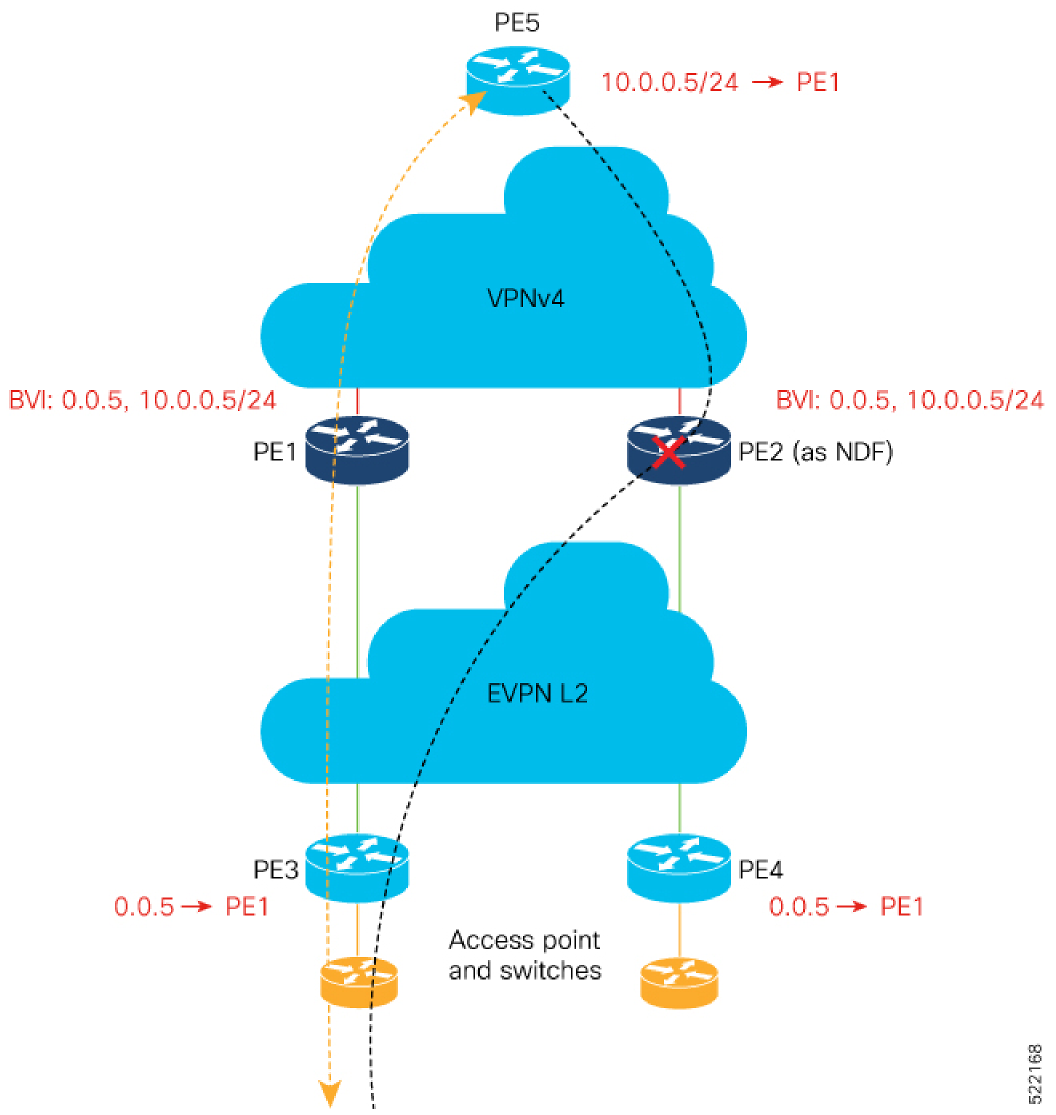
- Unicast Traffic – IPv4/IPv6 on BVI, Global VRF, customer VRF
- Core Isolation
- Cost-in/Cost-out
- DHCP Relay Agent
- MAC Mobility
- Convergence – with Unicast traffic

How Does Centralized Anycast Gateway Work?

EVPN performs DF election between centralized gateways to elect DF and non-DF node at EVI level for load balancing across subnets.

This example shows the centralized gateway without locally attached interface running in a single-active multi-homing mode.

Figure 100: Topology



In this topology, PE1 and PE2 are the centralized gateways where IRB interfaces are configured. Centralized gateways perform EVPN DF election based on EVI/ESI. PE1 is elected as DF whereas PE2 is elected as non-DF.

The following describes the centralized anycast gateway advertisement:

- PE1 is DF and PE2 is non-DF.

- PE1 advertises its EVI to remote PE (PE3 and PE4) where it gets installed as remote entry in the MAC forwarding database. PE2 does not install PE1 as remote MAC.
- PE1 and PE2 advertise their route to the remote PEs (PE3 and PE4).
- PE2 blocks all traffic from EVI to BVI interface.
- MAC-IP addresses of hosts are learned only at the IRB termination point, which is the centralized gateway. PE1 and PE2 advertise host MAC-IP to synchronize with the access EVI.

The following describes how the inter subnet communication happens:

- Host attached to PE3 sends an ARP request to EVPN L2 fabric.
- PE3 learns the host MAC as local and advertises its reachability using EVPN Route-Type2 to remote PEs, including gateways.
- The ARP request reaches PE1 through dataplane and gets blocked on PE2.
- PE1 learns the ARP request locally as the first-hop router.
- PE1 synchronizes ARP entry with PE2 to complete the adjacency.
- PE1 and PE2 receive corresponding host MAC route from BGP and installs it as remote MAC.
- PE1 responds with appropriate ARP reply to source host through PE3.
- PE3 sends unicast traffic to PE1.

The following describes how the intra subnet communication with a host connected to PE4 happens:

- EVPN layer-2 bridging capability where MAC reachability is learned over EVPN Route-Type2.
- Host H3 (attached to PE3) sends direct ARP request to the host IP connected to PE4.
- The PE4 host responds with an ARP reply using the unicast MAC address.

Restrictions

- Physical and bundle AC interfaces are not supported on centralized gateway. Configuration is not prevented but a warning syslog is generated and traffic from locally attached ACs is not supported.
- Only single-active multihoming mode is supported.
- Locally attached interface on gateway is supported only when the access-EVI is configured as Virtual Interface (VES) and EVPN non-DF blocking scheme is used.
- IRB interfaces on peering PE must be configured with anycast gateway.

Configuration

Perform this task to configure centralized gateway. You must have the same configuration on PE1 and PE2:

```
/* EVPN configuration */

Router(config)# evpn
Router(config-evpn)# virtual access-evi
Router(config-evpn-ac-evi)# ethernet-segment
Router(config-evpn-ac-evi-es)# identifier type 0 00.00.00.34.34.34.34.34.34
Router(config-evpn-ac-evi-es)# exit

Router(config-evpn-ac-evi)# core-isolation-group 1
Router(config-evpn-ac-evi)# exit
Router(config-evpn)# exit

/* L2VPN configuration for centralized gateway */

Router(config)# l2vpn
Router(config-l2vpn)#bridge group evpn_access_evi_8
Router(config-l2vpn-bg)#bridge-domain bd_8001
Router(config-l2vpn-bg-bd)#access-evi 8001
Router(config-l2vpn-bg-bd)#routed interface BVI8001
Router(config-l2vpn-bg-bd-bvi)#exit
Router(config-l2vpn-bg-bd)#exit
Router(config-l2vpn-bg)#exit
Router(config-l2vpn)#exit

/* BVI configuration */

Router(config)#Interface BVI8001
Router(config-if)#ipv4 address 10.1.1.1/24
Router(config-if)#mac-address 00aa.8001.00aa
Router(config-if)#commit
```

Show running configuration for centralized gateway:

```
interface BVI8001
 ipv4 address 10.1.1.1 255.255.255.0
 mac-address aa.8001.aa
!
evpn
 virtual access-evi
  ethernet-segment
   identifier type 0 00.00.00.34.34.34.34.34.34
  !
  core-isolation-group 1
 !
 !
l2vpn
 bridge group evpn_access_evi_8
  bridge-domain bd_8001
   access-evi 8001
   routed interface BVI8001
   !
  !
 !
!
```

Perform this task to configure PE2 and PE3:

Note Regular L2 EVPN configuration on PE2 and PE3 and BVI configurations aren't required.

```
Router#configure
Router(config)#evpn
Router(config-evpn)#evi 8001
Router(config-evpn-instance)#advertise-mac
Router(config)#interface bundle-ether80.1 l2transport
Router(config-subif)#encapsulation dot1q 0
Router(config-subif)#rewrite ingress tag pop 1 symmetric
Router(config-subif)#exit

Router(config)#l2vpn
Router(config-l2vpn)#bridge group evpn_access_evi_8
Router(config-l2vpn-bg)#bridge-domain bd_8001
Router(config-l2vpn-bg-bd)#interface bundle-ether 80.1
Router(config-l2vpn-bg-bd-ac)#evi 8001
Router(config-l2vpn-bg-bd-evi)#
```

Verification

Verify the configuration on PE1 and PE2 using the following show commands:

```
Router-PE1# show evpn ethernet-segment carving detail
Ethernet Segment Id      Interface                          Nexthops
------------------------ ---------------------------------- --------------------
0000.0000.3434.3434.3434 Access-EVI:all                     192.168.30.1
                                                            192.168.40.1
Main port         :
     Interface name : Access-EVI/all
Topology          :
Configured     : Single-active (AApS) (default)
  Service Carving   : Auto-selection
Service Carving Results:
     Forwarders     : 200
     Elected        : 100
Not Elected     : 100
```

ARP ND sync for DF and non-DF:

For DF, the state should be `dynamic` and for non-DF, it shows `EVPN-SYNC`.

```
/*For DF:*/
Router-PE1# show arp bvi 9002
Address         Age        Hardware Addr   State      Type  Interface
90.1.2.1        -          0011.0090.1111  Interface  ARPA  BVI9002
90.1.2.34       00:00:46   0012.9001.0002  Dynamic    ARPA  BVI9002

/* For NON-DF*/

Router-PE2# show arp bvi 9002
Address         Age        Hardware Addr   State      Type  Interface
90.1.2.1        -          0011.0090.1111  Interface  ARPA  BVI9002
90.1.2.34       -          0012.9001.0002  EVPN_SYNC  ARPA  BVI9002
```

BVI-Coupled Mode

When ACs go down, the BVI also goes down. However, with this mode enabled, the state of the BVI remains Up even though the ACs go down. Hence, the BVI state becomes EVPN-aware.

BVI tracks the Up or Down state of ACs and PWs in a bridge. When the EVPN port is available, there may be an L2 redirect path over EVI to carry the traffic between L3 to L2. However, this depends on the remote or peer EVI-EAD routes received.

Under certain conditions, you can reduce the churns of BVI state adjacency by keeping the BVI state Up. BVI state drives the state of EVPN_SYNC adjacencies being pushed to forwarding entries, thereby reducing the churns further. Keeping the BVI state Up, the router creates adjacencies in the forwarding table, which indicates that a local adjacency is invalid when an interface is down.

Configure BVI-Coupled Mode

Perform this task to configure BVI-coupled mode.

```
evpn
 evi 101
  bgp
   route-target import 60000:101
   route-target export 60000:101
  !
  bvi-coupled-mode

l2vpn
 bridge group BG-1
  bridge-domain BD-1
   interface Bundle-Ether100.101
   !
   routed interface BVI101
   !
   evi 101
```

Verification

Verify that the BVI-coupled mode is enabled.

```
Router# show evpn evi detail

VPN-ID     Encap      Bridge Domain                Type
---------- ---------- ---------------------------- -------------------
101        MPLS       BD-1                         EVPN
   Stitching: Regular
   Unicast Label  : 35048
   Multicast Label: 33000
   Reroute Label: 0
   Flow Label: N
   Control-Word: Enabled
   E-Tree: Root
   Forward-class: 0
   Advertise MACs: Yes
   Advertise BVI MACs: No
   Aliasing: Enabled
   UUF: Enabled
   Re-origination: Enabled
   Multicast:
     Source connected   : No
     IGMP-Snooping Proxy: No
     MLD-Snooping Proxy : No
   BGP Implicit Import: Enabled
   VRF Name: cust1
   Preferred Nexthop Mode: Off
BVI Coupled Mode: Yes -------------------------> enabled
   BVI Subnet Withheld: ipv4 No, ipv6 No
   RD Config: none
   RD Auto  : (auto) 201.201.201.1:101
   RT Auto  : 60000:101
   Route Targets in Use           Type
```

```
------------------------------ -------------
60000:101                      Import
60000:101                      Export
---------------------- --------------------
```

VM Mobility Support

VM mobility is the ability of virtual machines to migrate between one server and another while retaining their existing MAC and IP addresses.

The following are the two key components in EVPN Route Type 2 that enable VM Mobility:

- Host MAC advertisement component that is imported into local bridge MAC table, and Layer 2 bridged traffic across the network overlay.
- Host IP advertisement component that is imported into the IP routing table in a symmetric IRB design, enables routed traffic across the network overlay.

The above-mentioned components are advertised together in a single MAC + IP host route advertisement. An additional MAC-only route could also be advertised.

The following behaviors of VM are supported. The VM can:

- retain existing MAC and acquire a new IP address
- retain existing IP address and acquire a new MAC
- retain both existing MAC and IP address

Note IRB solution supports VM mobility with IP+MAC pair. VM mobility move, with new IP to MAC, or new MAC to IP, is not supported.

Configuring EVPN IRB

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface Bundle-Ether 3
RP/0/RSP0/CPU0:router(config-if)# lacp system mac 1.1.1
RP/0/RSP0/CPU0:router(config-if)# exit


/* Configure EVPN L3VRF per DC tenant. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# vrf irb1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 1000:1
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 1000:1
RP/0/RSP0/CPU0:router(config-vrf-af)# exit


/* Configure Layer 2 attachment circuit (AC) from multichassis (MC) bundle interface, and
bridge-group virtual interface (BVI) per bridge domain. */
/* Note: When a VM migrates from one subnet to another (subnet stretching), apply the
```

```
following IRB configuration to both the EVPN PEs. *\

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# interface bvi 1001
RP/0/RSP0/CPU0:router(config-if)# host-routing
RP/0/RSP0/CPU0:router(config-if)# vrf irb1
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.10.0.4 255.255.255.0
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 172.16.0.1 secondary
RP/0/RSP0/CPU0:router(config-if)# mac-address 00aa.1001.00aa

/* Configure EVPN Layer 2 bridging service. Note: This configuration is performed in Layer
 2 gateway or bridging scenario. */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group 1
Router(config-l2vpn-bg)# bridge-domain 1-1
Router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/1.1
Router(config-l2vpn-bg-bd-ac)# evi 1
Router(config-l2vpn-bg-bd-ac-evi)# commit
Router(config-l2vpnbg-bd-ac-evi)# exit


/* Configure BGP. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 3107
RP/0/RSP0/CPU0:router(config-bgp)# vrf irb1
RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd auto
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute connected
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute static
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# exit


/* Configure EVPN, and configure main bundle ethernet segment parameters in EVPN. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# evi 2001
RP/0/RSP0/CPU0:router(config-evpn-evi)# bgp
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target import 1000:1
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# route-target export 1000:1
RP/0/RSP0/CPU0:router(config-evpn-evi-bgp)# exit
RP/0/RSP0/CPU0:router(config-evpn-evi)# advertise-mac
RP/0/RSP0/CPU0:router(config-evpn-evi)# unknown-unicast-suppression


/* Configure Layer 2 VPN. */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# l2vpn
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group irb
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain irb1
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# interface bundle-Ether3.1001
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-ac)# routed interface BVI100
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-bvi)# split-horizon group core
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd-bvi)# evi 10001
```

Running Configuration for EVPN IRB

```
/* Configure LACP */
interface Bundle-Ether3
 lacp system mac 1.1.1
!



/* Configure EVPN Layer 3 VRF per DC tenant. */
vrf irb1
address-family ipv4 unicast
  import route-target
   1000:1
  !
  export route-target
   1000:1
  !

!
!

/* Configure Layer 2 attachment circuit (AC) from multichassis (MC) bundle interface, and
bridge-group virtual interface (BVI) per bridge domain./*


interface Bundle-Ether3.1001 l2transport
 encapsulation dot1q 1001
 rewrite ingress tag pop 1 symmetric
!
interface BVI1001
 host-routing
 vrf irb1
 ipv4 address 10.0.1.1 255.255.255.0
 mac-address 0000.3030.1
!

/* Configure BGP. */

router bgp 3107
 vrf irb1
  rd auto
  address-family ipv4 unicast
  redistribute connected
  redistribute static
!
!

/* Configure EVPN. */

evpn
evi 10001
  bgp
   route-target import 1000:1
   route-target export 1000:1
  !
  advertise-mac
  unknown-unicast-suppression
!

/* Configure Layer2 VPN. */
```

```
l2vpn
bridge group irb
  bridge-domain irb1
   interface Bundle-Ether3.1001
   !
   routed interface BVI1001
    split-horizon group core
   !
   evi 10001
   !
  !
```

Verify EVPN IRB

Verify the Address Resolution Protocol (ARP) protocol entries, and synced entries in multi-homing scenarios.

```
RP/0/RSP0/CPU0:router# show arp vrf evpn1

-----------------------------------------------------------------
0/1/CPU0
-----------------------------------------------------------------
Address    Age        Hardware Addr   State      Type   Interface

10.1.1.1    -         0010.0001.0001  Interface  ARPA   BVI1
10.1.1.11 02:23:46    1000.0001.0001  Dynamic    ARPA   BVI1
10.1.1.93   -         0000.f65a.357c  EVPN_SYNC  ARPA   BVI1
10.1.2.1    -         0011.0112.0001  Interface  ARPA   BVI2
10.1.2.91 02:24:14    0000.f65a.3570  Dynamic    ARPA   BVI2
10.1.2.93 02:21:52    0000.f65a.357d  Dynamic    ARPA   BVI2
---------------------------------------------------------------
0/0/CPU0
---------------------------------------------------------------
Address   Age         Hardware Addr   State      Type  Interface

10.1.1.1  -           0010.0001.0001  Interface  ARPA  BVI1
10.1.1.11 02:23:46    1000.0001.0001  Dynamic    ARPA  BVI1
10.1.1.93 -           0000.f65a.357c  EVPN_SYNC  ARPA  BVI1
10.1.2.1  -           0011.0112.0001  Interface  ARPA BVI2
10.1.2.91 02:24:14    0000.f65a.3570  Dynamic    ARPA BVI2
10.1.2.93 02:21:52    0000.f65a.357d  Dynamic    ARPA BVI2
```

Verify the adjacency entries, particularly verify newly added information for synced IPv4 and IP ARP entries.

```
RP/0/RSP0/CPU0:router# show adjacency ipv4 BVI 1 internal detail location 0/0/CPU0

BVI1, 10.1.1.93 (ipv4)
Version: 1169, references: 2, transient lock: 0
Encapsulation information (14 bytes) 0000f65a357c0000f65a357c0800 MTU: 1500
 Adjacency pointer is: 0x770a9278
 Platform adjacency pointer is: 0x7d7bc380
 Last updated: Feb 28 15:58:21.998
 Adjacency producer: arp (prod_id: 10)
 Flags: incomplete adj,
 Additional Adjacency Information (4 bytes long),
 Upto first 4 bytes (in hex): 01000000
 Netio idb pointer not cached Cached interface type: 78
```

```
Adjacency references:
bfd_agent (JID 150, PID 3637), 0 reference
l2fib_mgr (JID 185, PID 4003), 0 reference
fib_mgr (JID 294, PID 3605), 1 reference
aib (JID 314, PID 3590), 1 reference

BVI1, 10.1.1.11 (ipv4) Version: 1493,
references: 3, transient lock: 0
Encapsulation information (14 bytes) 1000000100010010000100010800
MTU: 1500
Adjacency pointer is: 0x770ab778
Platform adjacency pointer is: 0x7d7bcb10
Last updated: Mar 2 17:22:00.544
Adjacency producer: arp (prod_id: 10)
Flags: incomplete adj,
Netio idb pointer not cached Cached interface type: 78
Adjacency references:
bfd_agent (JID 150, PID 3637), 0 reference
l2fib_mgr (JID 185, PID 4003), 1 reference
fib_mgr (JID 294, PID 3605), 1 reference
aib (JID 314, PID 3590), 1 reference
```

Verify the entries to obtain details learnt in L2FIB line cards. In multi-homing active-active scenario, the link-local addresses are also updated and distributed to EVPN peer gateways.

```
RP/0/RSP0/CPU0:router# show l2vpn mac-learning mac-ipv4 all location 0/0/cPU0

Topo ID  Producer  Next Hop(s)  Mac Address     IP Address

6        0/0/CPU0   BV1        1000.0001.0001      10.1.1.11
7        0/0/CPU0   BV2        0000.f65a.3570      10.1.2.91
7       0/0/CPU0    BV2        0000.f65a.357d      10.1.2.93

RP/0/RSP0/CPU0:router# show l2vpn mac-learning mac-ipv4 all location 0/0/cPU0

Topo ID  Producer  Next Hop(s)  Mac Address    IP Address

6      0/0/CPU0    BV1       0000.f65a.357c    fe80::200:f6ff:fe5a:357c
7      0/0/CPU0    BV2       0000.f65a.3570    10:1:2::91
7      0/0/CPU0    BV2       0000.f65a.357d    10:1:2::93
7      0/0/CPU0    BV2       0000.f65a.3570    fe80::200:f6ff:fe5a:3570
```

Verify sequence ID for VM mobility.

```
RP/0/RSP0/CPU0:router# show l2route evpn mac-ip all detail

Sun Apr 30 18:09:19.368 PDT
Flags: (Stt)=Static; (L)=Local; (R)=Remote; (F)=Flood;
(N)=No Redistribution; (Rtr)=Router MAC; (B)=Best Route;
(P)=Probe; (S)=Peer Sync; (F)=Flush;
(D)=Duplicate MAC; (Z)=Frozen MAC;

Topo ID    Mac Address     IP Address  Prod   Next Hop(s)         Seq No  Flags
Opaque Data Type    Opaque Data Len   Opaque Data Value
-------    -----------     ----------  ----   ----------          ------  -----
----------------    ---------------   -----------------
33         0022.6730.0001  10.130.0.2  L2VPN  Bundle-Ether6.1300  0      SB 0 12
0x06000000           0x22000080        0x00000000
```

```
Last Update: Sun Apr 30 15:00:01.911 PDT

33         0022.6730.0002 10.130.0.3  LOCAL  Bundle-Ether6.1300   0       B           N/A
                 N/A              N/A
```

Verify the entries to obtain details learnt in L2FIB RP when it is an aggregator. Route processor (RP) entries are aggregated entries obtained from the line cards. In some cases of MAC move, there could be different states for the same MAC. This is displayed in RP aggregated entries. RP determines the update to be sent to L2RIB according to MAC-Learning algorithms.

```
RP/0/RSP0/CPU0:router#  show l2vpn mac-learning mac-ipv4 all location 0/RSP0/CPU0

Topo ID  Producer        Next Hop(s)        Mac Address         IP Address
-------      --------     -----------        --------------      ----------
6            0/0/CPU0       BV1            1000.0001.0001      10.1.1.11
7            0/0/CPU0       BV2            0000.f65a.3570      10.1.2.91
7            0/0/CPU0       BV2            0000.f65a.357d      10.1.2.93
```

Verify the entries in L2RIB that are updated by RP L2FIB. Note the following when you verify the entries:

- The entries with producer as L2VPN and NH as remote IP are learnt from the remote peer gateways, which are learnt from BGP, updated to EVPN, and then updated to L2RIB. So these entries are not from local IP-MAC learning.
- The entries with producer as L2VPN and NH as local bundle interfaces are synced entries from MH-AA peer gateway.
- The entries with producer as LOCAL and NH as local bundle interfaces are dynamically learnt local entries.

```
RP/0/RSP0/CPU0:router# show l2route evpn mac-ip evi 6

Topo ID      Mac Address        IP Address               Prod        Next Hop(s)
--------     --------------     ---------------          ------      -------------
6            0000.f65a.3569     10.1.1.101               L2VPN     172.16.0.2/24014/ME
6            0000.f65a.3575     10.1.1.97                L2VPN     172.16.0.7/24025/ME
6            0000.f65a.3575     10:1:1::97               L2VPN     172.16.0.7/24025/ME
6            0000.f65a.3575     fe80::200:f6ff:fe5a:3575 L2VPN     172.16.0.7/24025/ME
6            0000.f65a.357c     10.1.1.93                L2VPN      Bundle-Ether1.11
6            0000.f65a.357c     10:1:1::93               L2VPN      Bundle-Ether1.11
6            0000.f65a.357c     fe80::200:f6ff:fe5a:357c LOCAL      Bundle-Ether1.11
6            0010.0001.0012     10.1.1.12                L2VPN      172.16.0.7/24025/ME
6            1000.0001.0001     10.1.1.11                LOCAL      Bundle-Ether1.11
6            90e2.ba8e.c0c9     10.1.1.102               L2VPN      172.16.0.2/24014/ME
```

Verify entries to obtain details of EVPN.

```
RP/0/RSP0/CPU0:router# show evpn evi vpn-id 1 mac ipv4 10.1.1.93 detail

EVI       MAC address         IP address            Nexthop             Label
```

```
----      ---------------     ----------          ----------          -----
1          0000.f65a.357c      10.1.1.93           172.16.0.2          24014

Ethernet Tag : 0
Multi-paths Resolved : True
Static : No
Local Ethernet Segment : N/A
Remote Ethernet Segment : 0100.6cbc.a77c.c180.0000
Local Sequence Number : N/A
Remote Sequence Number : 0
Local Encapsulation : N/A
Remote Encapsulation : MPLS
```

Verify local BGP entries with appropriate second label and second IP VRF route-target.

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn rd 172.16.0.1:1
[2][0][48][0000.f65a.357c][32][10.1.1.93]/136

BGP routing table entry for [2][0][48][0000.f65a.357c][32][10.1.1.93]/136, Route
Distinguisher: 172.16.0.1:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 3772 3772
Local Label: 24013
Last Modified: Feb 28 16:06:37.073 for 2d19h
Paths: (2 available, best #1)
Advertised to peers (in unique update groups):
172.16.0.9
Path #1: Received by speaker 0
Advertised to peers (in unique update groups):
172.16.0.9
Local
0.0.0.0 from 0.0.0.0 (172.16.0.1)
Second Label 24027                              >>>>  Second label when IRB host-routing
 is enabled.
Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
rib-install
Received Path ID 0, Local Path ID 0, version 3772
Extended community: SoO:172.16.0.2:1 RT:100:100
EVPN ESI: 0100.6cbc.a77c.c180.0000
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 (metric 101) from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24031
Origin IGP, localpref 100, valid, internal, add-path, import-candidate, imported, rib-install
Received Path ID 0, Local Path ID 2, version 3769
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100  >>>  Second RT is IP VRF RT for
remote to import into IP VRF routing table.
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn rd 172.16.0.1:1
[2][0][48][0000.f65a.357c][128][10:1:1::93]/232

[2][0][48][0000.f65a.357c][128][10:1:1::93]/232
BGP routing table entry for [2][0][48][0000.f65a.357c][128][10:1:1::93]/232, Route
```

```
Distinguisher: 172.16.0.1:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 3172 3172
Local Label: 24013
Last Modified: Feb 28 11:34:33.073 for 3d00h
Paths: (2 available, best #1)
Advertised to peers (in unique update groups):
172.16.0.9
Path #1: Received by speaker 0
Advertised to peers (in unique update groups):
172.16.0.9
Local
0.0.0.0 from 0.0.0.0 (172.16.0.1)
Second Label 24029
Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
rib-install
Received Path ID 0, Local Path ID 0, version 3172
Extended community: SoO:172.16.0.2:1 RT:100:100
EVPN ESI: 0100.6cbc.a77c.c180.0000
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 (metric 101) from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24033
Origin IGP, localpref 100, valid, internal, add-path, import-candidate, imported, rib-install
Received Path ID 0, Local Path ID 2, version 3167
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify the remote peer gateway BGP entries with correct label and route-target. Particularly verify the local auto-generated RD on a remote EVPN gateway. EVPN type-2 routes are imported into EVPN. The host routes of IPv4 /32 addresses are imported only into IP VRF route-table in the remote EVPN gateway, but not in the local EVPN gateway where local BVI adjacency is used to overwrite RIB entries.

```
RP/0/RSP0/CPU0:router#  show bgp l2vpn evpn rd 172.16.0.7:1
[2][0][48][0000.f65a.357c][32][10.1.1.93]/136
BGP routing table entry for [2][0][48][0000.f65a.357c][32][10.1.1.93]/136, Route
Distinguisher: 172.16.0.7:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 16712 16712
Last Modified: Feb 28 16:06:36.448 for 2d19h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24013, Second Label 24027 >>>> First label for L2 MAC unicast bridging;
second label for EVPN IRB host-routing
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 0, version 16712
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
```

```
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24031
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 1, version 16706
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

```
RP/0/RSP0/CPU0:router# show bgp l2vpn evpn rd 172.16.0.7:1
[2][0][48][0000.f65a.357c][128][10:1:1::93]/232

BGP routing table entry for [2][0][48][0000.f65a.357c][128][10:1:1::93]/232, Route
Distinguisher: 172.16.0.7:1
Versions:
Process bRIB/RIB SendTblVer
Speaker 6059 6059
Last Modified: Feb 28 12:03:22.448 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24013, Second Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 0, version 6043
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24014, Second Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported,
rib-install
Received Path ID 0, Local Path ID 1, version 6059
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
EVPN ESI: 0100.6cbc.a77c.c180.0000
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify the remote peer gateway with host routes of IPv4 /32 addresses imported into the IP VRF routing table.

```
RP/0/RSP0/CPU0:router#  show bgp vpnv4 unicast vrf evpn1 10.1.1.93/32

BGP routing table entry for 10.1.1.93/32, Route Distinguisher: 172.16.0.7:11
Versions:
```

```
Process bRIB/RIB SendTblVer
Speaker 22202 22202
Last Modified: Feb 28 16:06:36.447 for 2d19h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24027
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22202
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1 >>>>
 The source from

   >>>> L2VPN and from

   >>>> synced ARP entry.
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24031
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22201
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 17.0.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1 >>>>
 The source from

   >>>> L2VPN and

   >>>> from dynamic

   >>>> ARP entry.
```

```
RP/0/RSP0/CPU0:router# show bgp vpnv6 unicast vrf evpn1 10:1:1::93/128

BGP routing table entry for 10:1:1::93/128, Route Distinguisher: 172.16.0.7:11
Versions:
Process bRIB/RIB SendTblVer
Speaker 22163 22163
Last Modified: Feb 28 12:09:30.447 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22163
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1 >>>>
 Source from

   >>>> L2VPN and from
   >>>> synced ARP entry.
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22163
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1 >>>>
 Source from

   >>>> L2VPN and from

   >>>> dynamic ARP entry.
```

```
RP/0/RSP0/CPU0:router# show bgp vpnv6 unicast vrf evpn1 10:1:1::93/128

BGP routing table entry for 10:1:1::93/128, Route Distinguisher: 172.16.0.7:11
Versions:
Process bRIB/RIB SendTblVer
Speaker 22163 22163
Last Modified: Feb 28 12:09:30.447 for 2d23h
Paths: (2 available, best #1)
Not advertised to any peer
Path #1: Received by speaker 0
Not advertised to any peer
Local
172.16.0.1 from 172.16.0.9 (172.16.0.1)
Received Label 24029
Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate, imported
Received Path ID 0, Local Path ID 0, version 22163
Extended community: SoO:172.16.0.2:1 RT:100:1 RT:100:100
Originator: 172.16.0.1, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.1:1
Path #2: Received by speaker 0
Not advertised to any peer
Local
172.16.0.2 from 172.16.0.9 (172.16.0.2)
Received Label 24033
Origin IGP, localpref 100, valid, internal, backup, add-path, import-candidate, imported
Received Path ID 0, Local Path ID 1, version 22163
Extended community: SoO:172.16.0.2:1 RT:200:1 RT:700:100
Originator: 172.16.0.2, Cluster list: 172.16.0.9
Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 172.16.0.2:1
```

Verify local forwarding with local adjacency which overwrite the RIB entries, and remote peer that use the IP VRF host route entries for IP VPN forwarding.

```
RP/0/RSP0/CPU0:router#  show bgp vpnv4 unicast vrf evpn1 10.1.1.93/32

-- For local routing and forwarding
RP/0/RSP0/CPU0:PE11-R1#show route vrf evpn1 10.1.1.93
Routing entry for 10.1.1.93/32
Known via "bgp 3107", distance 200, metric 0, type internal
Installed Feb 28 15:57:28.154 for 2d20h
```

```
Routing Descriptor Blocks
172.16.0.2, from 172.16.0.9      >>>  From MH-AA peer.
Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
No advertising protos.

RP/0/RSP0/CPU0:PE11-R1# show cef vrf evpn1 10.1.1.93 location 0/0/CPU0
10.1.1.93/32, version 0, internal 0x1120001 0x0 (ptr 0x7b40052c) [1], 0x0 (0x7b286010), 0x0
 (0x0)
Updated Feb 28 15:58:22.688
local adjacency 10.1.1.93
Prefix Len 32, traffic index 0, Adjacency-prefix, precedence n/a, priority 15
via 10.1.1.93/32, BVI1, 2 dependencies, weight 0, class 0 [flags 0x0]
path-idx 0 NHID 0x0 [0x7f531f88 0x0]
next hop
local adjacency              >>> Forwarding with local synced ARP adjacency entries.


For remote routing and forwarding:

RP/0/RSP0/CPU0:router# show route vrf evpn1 10.1.1.93

Routing entry for 10.1.1.93/32
Known via "bgp 3107", distance 200, metric 0
Number of pic paths 1 , type internal
Installed Feb 28 16:06:36.431 for 2d20h
Routing Descriptor Blocks
172.16.0.1, from 172.16.0.9
Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
172.16.0.2, from 172.16.0.9, BGP backup path
Nexthop in Vrf: "default", Table: "default", IPv4 Unicast, Table Id: 0xe0000000
Route metric is 0
No advertising protos.

RP/0/RSP0/CPU0:router# show cef vrf evpn1 10.1.1.93 location 0/0/CPU0

10.1.1.93/32, version 86, internal 0x5000001 0x0 (ptr 0x99fac884) [1], 0x0 (0x0), 0x208
(0x96c58494)
Updated Feb 28 16:06:39.285
Prefix Len 32, traffic index 0, precedence n/a, priority 3
via 172.16.0.1/32, 15 dependencies, recursive [flags 0x6000]
path-idx 0 NHID 0x0 [0x97955380 0x0]
recursion-via-/32
next hop VRF - 'default', table - 0xe0000000
next hop 172.16.0.1/32 via 34034/0/21
next hop 100.0.57.5/32 Te0/0/0/3 labels imposed {ImplNull 24011 24027}
next hop 100.0.67.6/32 Te0/0/0/1 labels imposed {ImplNull 24009 24027}
via 172.16.0.2/32, 11 dependencies, recursive, backup [flags 0x6100]
path-idx 1 NHID 0x0 [0x979554a0 0x0]
recursion-via-/32
next hop VRF - 'default', table - 0xe0000000
next hop 172.16.0.2/32 via 34035/0/21
next hop 100.0.57.5/32 Te0/0/0/3 labels imposed {ImplNull 24012 24031}
next hop 100.0.67.6/32 Te0/0/0/1 labels imposed {ImplNull 24010 24031}
```

The following sections describe how to verify the subnet stretching.

Verify the VRF.

```
RP/0/RP0/CPU0:leafW# show run vrf cust130
```

```
vrf cust130
address-family ipv4 unicast
  import route-target
   130:130
  !
  export route-target
   130:130
  !
!
!
```

Verify the BGP configuration.

```
 RP/0/RP0/CPU0:leafW# show run router bgp | begin vrf cust130

vrf cust130
  rd auto
  address-family ipv4 unicast
   label mode per-vrf
   maximum-paths ibgp 10
   redistribute connected
  !
!
```

Verify the L2VPN.

```
RP/0/RP0/CPU0:leafW# show run l2vpn bridge group bg130

l2vpn
bridge group bg130
  bridge-domain bd130
   interface Bundle-Ether1.1300
   !
   interface Bundle-Ether5.1300
   !
   routed interface BVI130
   evi 130
   !
  !
!
!
```

EVPN IPv6 Hosts with Mobility

EVPN IPv6 Hosts with Mobility feature enables you to provide EVPN IPv6 service over IPv4-MPLS core network. This feature supports all-active multihoming and virtual machine (VM) or host move.

Service Providers (SPs) use a stable and established core with IPv4-MPLS backbone for providing IPv4 VPN services. The IPv6 VPN Provider Edge Transport over MPLS (IPv6 on Provider Edge Routers [6PE] and IPv6 on VPN Provider Edge Routers [6VPE]) facilitates SPs to offer IPv6 VPN services over IPv4 backbone without an IPv6 core. The provide edge (PE) routers run MP-iBGP to advertise IPv6 reachability and IPv6 label distribution. For 6PE, the labels are allocated per IPv6 prefix learnt from connected customer edge (CE) routers and for 6VPE, the PE router can be configured to allocate labels on a per-prefix or per-CE and per-VRF level.

Mobility Support

In global VRF, mobility is not supported. However, you can move a host from one ES to another ES within the same bridge domain. The host gets a new MAC address and IP address. The host can have multiple IP addresses for the same MAC address.

In non-default VRF, mobility is supported with the following conditions:

- Basic MAC move: The IP address and MAC address remains the same. You can move a host from one ES to another ES with the same IP address and MAC address.
- Same MAC address but with a different IP address: The host gets a new IP address
- Same IP address but with a different MAC address: The host gets a new MAC address but retains the same IP address
- Multiple IP addresses with the same MAC address: Many VMs are involved in the same the MAC move

Restrictions

- In customer VRFs, when host routing is not configured, MAC-IP advertisement is different between zero ESI and none-zero ESI. When host routing is not configured, MAC-IP with non-zero ESI is advertised without L3 RT (VRF RT). MAC-IP with zero ESI is not advertised. The following table lists the behavior of MAC-IP advertisement with respect to ESI and host routing.

ESI Type	With host routing	Without host routing
MAC-IP with non-zero ESI	Advertised with L3 VRF RT	Advertised without L3 VRF RT
MAC-IP with zero ESI	Advertised with L3 VRF RT	Not advertised

- In global VRF, Layer 2 stretch is not supported.
- MAC move in global VRF is only supported if the host is within the same bridge domain. You can move a host from one ES to another ES within the same bridge domain.
- Duplication of IP address detection is not supported.
- Maximum number of leafs allowed per ESI is two.

Configure EVPN IPv6 Hosts with Mobility

Perform the following tasks to configure EVPN IPv6 Hosts with Mobility feature:

- Configure VRF
- Configure ISIS
- Configure BGP
- Configure AC interface
- Configure BVI interface
- Configure EVPN
- Configure L2VPN

Note

- You cannot configure the EVPN remote peer using the VPNv4 unicast if you have configured the **advertise vpnv4 unicast re-originated** command under the L2VPN EVPN address-family. You can either configure the VPNv4 unicast or the advertise vpnv4 unicast re-originated under L2VPN EVPN address-family.
- You cannot configure the EVPN remote peer using the VPNv6 unicast if you have configured the **advertise vpnv6 unicast re-originated** command under the L2VPN EVPN address-family. You can either configure the VPNv6 unicast or the advertise vpnv6 unicast re-originated under L2VPN EVPN address-family.

```
/* Configure VRF */

Router# configure
Router(config)# vrf cust102
Router(config-vrf)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target 160102:16102
Router(config-vrf-af)# export route-target 160102:16102
Router(config-vrf-af)# exit
!
Router(config-vrf)# address-family ipv6 unicast
Router(config-vrf-af)# import route-target 6160102:16102
Router(config-vrf-af)# export route-target 6160102:16102
Router(config-vrf-af)# commit
!

/* Configure ISIS */

Router# configure
Route(config)# router isis v6
Route(config-isis)# 49.0001.0000.0160.0005.00
Route(config-isis)# nsr
Route(config-isis)# log adjacency changes
Route(config-isis)# lsp-gen-interval maximum-wait 5000 initial-wait 1 secondary-wait
20
Route(config-isis)# lsp-mtu 1468
Route(config-isis)# lsp-refresh-interval 65000
Route(config-isis)# max-lsp-lifetime 65535
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# microloop avoidance protected
Route(config-isis-af)# spf-interval maximum-wait 5000 initial-wait 1 secondary-wait 20
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface Bundle-Ether10
Route(config-isis-if)# point-to-point
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# fast-reroute per-prefix
Route(config-isis-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-af)# metric 10
Route(config-isis-af)# exit
!
Route(config-isis)# interface Bundle-Ether20
Route(config-isis-if)# point-to-point
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# fast-reroute per-prefix
Route(config-isis-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-af)# metric 10
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# passive
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback10
Route(config-isis-if)# passive
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 1605
Route(config-isis-af)# commit
Route(config-isis-af)# exit
!

/* Configure Segment Routing */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# commit

/* Configure BGP */

Router(config)# router bgp 100
Router(config-bgp)# bfd minimum-interval 50
Router(config-bgp)# bfd multiplier 3
Router(config-bgp)# bgp router-id 160.0.0.5
Router(config-bgp)# address-family ipv4 unicast      --->  To support V4 Global VRF
Router(config-bgp-af)# maximum-paths ibgp 10 unequal-cost  ---> ECMP
Router(config-bgp-af)# redistribute connected    --> V4 Global VRF
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family ipv4 unicast      --->  VRF
Router(config-bgp-af)# vrf all
Router(config-bgp-af)# label mode per-vrf
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family ipv6 unicast   ---> For 6PE
Router(config-bgp-af)# label mode per-vrf
Router(config-bgp-af)# maximum-paths ibgp 8
Router(config-bgp-af)# redistribute static
Router(config-bgp-af)# allocate-label all
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family vpnv6 unicast   ---> 6 VPE
Router(config-bgp-af)# vrf all
Router(config-bgp-af)# label mode per-vrf
Router(config-bgp-af)# exit
!
Router(config-bgp)# address-family l2vpn evpn   ----> EVPN
Router(config-bgp-af)# bgp implicit-import      ----> Global VRF
Router(config-bgp-af)# exit
!
Router(config-bgp)# neighbor-group evpn-rr
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# bfd fast-detect
Router(config-bgp-nbr)# update-source loopback0
Router(config-bgp-nbr)# address-family ipv4 unicast
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy nh-lo10 out
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp-nbr)# address-family ipv6 labeled-unicast  ----> For 6PE
Router(config-bgp-nbr-af)# route-policy pass-all out
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy pass-all in
Router(config-bgp-nbr-af)# route-policy nh-lo10 out
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated -> For Route Type 5
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated -> For Route Type 5
Router(config-bgp-nbr-af)# exit
!
Router(config-bgp)# neighbor 160.0.0.1
Router(config-bgp-nbr)# use neighbor-group evpn-rr
Router(config-bgp-nbr)# exit
!
Router(config-bgp)# neighbor 160.0.0.2
Router(config-bgp-nbr)# use neighbor-group evpn-rr
Router(config-bgp-nbr)# exit
!
Router(config-bgp)# vrf all
Router(config-bgp-vrf)# rd 1605:102
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-vrf-af)# redistribute connected   --->  Triggers Route Type 5
Router(config-bgp-vrf-af)# exit
!
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# label mode per-vrf
Router(config-bgp-vrf-af)# maximum-paths ibgp 10 unequal-cost
Router(config-bgp-vrf-af)# redistribute connected
Router(config-bgp-vrf-af)# exit
!

/* Configure AC interface */

Router(config)# interface Bundle-Ether1.102 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 102
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit
Router(config-l2vpn-subif)# exit

/* Configure BVI interface */

Router(config)# interface BVI100
Router(config-if)# ipv4 address 56.78.100.1 255.255.255.0
Router(config-if)# ipv6 address 56:78:100::1/64
Router(config-if)# mac-address 22.22.22
Router(config-if)# exit
!
Router(config)# interface BVI102
Router(config-if)# host-routing
Router(config-if)# vrf cust102
Router(config-if-vrf)# ipv4 address 56.78.102.1 255.255.255.0
Router(config-if-vrf)# ipv6 nd dad attempts 0
Router(config-if-vrf)# ipv6 address 56:78:100::1/64
Router(config-if-vrf)# ipv6 address 56:78:102::1/64
Router(config-if-vrf)# mac-address 22.22.22
Router(config-if)# commit



/* Configure EVPN, and configure main bundle ethernet segment parameters in EVPN */

Router# configure
Router(config)# evpn
Router(config-evpn)# evi 102
Router(config-evpn-evi)# bgp
Router(config-evpn-evi)# rd 1605:102
Router(config-evpn-evi-bgp)# route-target import 160102:102
Router(config-evpn-evi-bgp)# route-target export 160102:102
Router(config-evpn-evi-bgp)# exit
Router(config-evpn-evi)# advertise-mac
Router(config-evpn-evi)# exit
!
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 56.56.56.56.56.56.56.56.01
Router(config-evpn-ac-es)# exit
!
Router(config-evpn)# interface Bundle-Ether2
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 56.56.56.56.56.56.56.56.02
Router(config-evpn-ac-es)# commit

/* Configure L2VPN */

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg102
Router(config-l2vpn-bg)# bridge-domain bd102
Router(config-l2vpn-bg-bd)# interface Bundle-Ether1.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether2.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether3.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether4.102
Router(config-l2vpn-bg-bd-ac)# exit
!
Router(config-l2vpn-bg-bd)# interface Bundle-Ether5.102
Router(config-l2vpn-bg-bd-ac)# routed interface BVI102
Router(config-l2vpn-bg-bd-bvi)# evi 102
Router(config-l2vpn-bg-bd-bvi-evi)# commit
```

Running Configuration

```
/* Configure VRF */

vrf cust102
 address-family ipv4 unicast
 import route-target
 160102:16102
 !
 export route-target
 160102:16102
 !
 !
 address-family ipv6 unicast
 import route-target
 6160102:16102
 !
 export route-target
 6160102:16102
 !
 !
!

/ * Configure ISIS */

router isis v6
 net 49.0001.0000.0160.0005.00
 nsr
 log adjacency changes
 lsp-gen-interval maximum-wait 5000 initial-wait 1 secondary-wait 20
 lsp-mtu 1468
 lsp-refresh-interval 65000
 max-lsp-lifetime 65535
 address-family ipv4 unicast
 metric-style wide
 microloop avoidance protected
 spf-interval maximum-wait 5000 initial-wait 1 secondary-wait 20
 segment-routing mpls sr-prefer
 segment-routing prefix-sid-map advertise-local
 !
 interface Bundle-Ether10
 point-to-point
 address-family ipv4 unicast
 fast-reroute per-prefix
 fast-reroute per-prefix ti-lfa
 metric 10
 !
 !
 interface Bundle-Ether20
 point-to-point
 address-family ipv4 unicast
 fast-reroute per-prefix
 fast-reroute per-prefix ti-lfa
 metric 10
 !
 !
 interface Loopback0
 passive
 address-family ipv4 unicast
 !
 !
 interface Loopback10
 passive
 address-family ipv4 unicast
 prefix-sid index 1605
 !
 !
!

/ * Configure Segment Routing */

segment-routing
 global-block 16000 23999
!
```

```
/ * Configure BGP */

router bgp 100
 bfd minimum-interval 50
 bfd multiplier 3
 bgp router-id 160.0.0.5
 address-family ipv4 unicast      --->  To support V4 Global VRF
  maximum-paths ibgp 10 unequal-cost  ---> ECMP
  redistribute connected    --> V4 Global VRF
 !
 address-family vpnv4 unicast ---> VRF
  vrf all
   label mode per-vrf
 !
 address-family ipv6 unicast   ---> For 6PE
  label mode per-vrf
  maximum-paths ibgp 8
  redistribute connected
  redistribute static
  allocate-label all
 !
 address-family vpnv6 unicast   ---> 6VPE
  vrf all
   label mode per-vrf
 !
 address-family l2vpn evpn   ----> EVPN
 bgp implicit-import         ----> Global VRF
 !

neighbor-group evpn-rr
 remote-as 100
 bfd fast-detect
 update-source Loopback0
 address-family ipv4 unicast
  route-policy pass-all in
  route-policy nh-lo10 out
 !
 address-family ipv6 labeled-unicast  ----> For 6PE
 route-policy pass-all out
 !
 address-family l2vpn evpn
 route-policy pass-all in
 route-policy nh-lo10 out
 advertise vpnv4 unicast re-originated   ---> For Route Type 5
 advertise vpnv6 unicast re-originated   ----> For Route Type 5
 !
 !
 neighbor 160.0.0.1
 use neighbor-group evpn-rr
 !
 neighbor 160.0.0.2
 use neighbor-group evpn-rr
 !
 vrf cust102
 rd 1605:102
 address-family ipv4 unicast
 label mode per-vrf
 maximum-paths ibgp 10 unequal-cost
 redistribute connected   <----- Triggers Route Type 5
 !
 address-family ipv6 unicast
 label mode per-vrf
 maximum-paths ibgp 10 unequal-cost
 redistribute connected
 !
 !

/* Configure AC interface */

interface Bundle-Ether1.102 l2transport
 encapsulation dot1q 102
 rewrite ingress tag pop 1 symmetric
!
/* Configure BVI interface */
interface BVI100
 ipv4 address 56.78.100.1 255.255.255.0
 ipv6 address 56:78:100::1/64
 mac-address 22.22.22
!
interface BVI102
 host-routing
 vrf cust102
 ipv4 address 56.78.102.1 255.255.255.0
 ipv6 nd dad attempts 0
 ipv6 address 56:78:100::1/64
 ipv6 address 56:78:102::1/64
 mac-address 22.22.22
!



/* Configure EVPN */

evpn
 evi 102
 bgp
 rd 1605:102
 route-target import 160102:102
 route-target export 160102:102
 !
 advertise-mac
 !
 !
!
interface Bundle-Ether1
 ethernet-segment
 identifier type 0 56.56.56.56.56.56.56.56.01
 !
 !
 interface Bundle-Ether2
 ethernet-segment
 identifier type 0 56.56.56.56.56.56.56.56.02
 !
 !

/* Configure L2VPN */

l2vpn
 bridge group bg102
 bridge-domain bd102
 interface Bundle-Ether1.102
 !
 interface Bundle-Ether2.102
 !
 interface Bundle-Ether3.102
 !
 interface Bundle-Ether4.102
 !
 interface Bundle-Ether5.102
 !
 routed interface BVI102
 !
 evi 102
 !
 !
 !
!
```

Verification

Verify that you have configured EVPN IPv6 Hosts with Mobility feature is configured.

```
/* 6PE and Static Route Advertisement */
Host route is advertised as EVPN Route Type 2

Router# show bgp ipv6 unicast 56:78:100::2
BGP routing table entry for 56:78:100::2/128
Versions:
 Process bRIB/RIB SendTblVer
 Speaker 212 212
 Local Label: 2
Last Modified: Oct 31 19:13:10.998 for 00:00:19
Paths: (1 available, best #1)
 Not advertised to any peer
 Path #1: Received by speaker 0
 Not advertised to any peer
 Local
 160.5.5.5 (metric 20) from 160.0.0.1 (160.0.0.5)
 Received Label 2
 Origin IGP, localpref 100, valid, internal, best, group-best, imported
 Received Path ID 0, Local Path ID 0, version 212
 Extended community: Flags 0x20: SoO:160.5.5.5:100 RT:160100:100
 mac: 00:06:01:00:01:02
 Originator: 160.0.0.5, Cluster list: 100.0.0.4
 Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 1605:100

/* Manually configured static route in global VRF */

Router# show bgp ipv6 unicast 56:78:100::2

BGP routing table entry for 30::1/128
Versions:
 Process bRIB/RIB SendTblVer
 Speaker 9 9
 Local Label: 2
Last Modified: Oct 30 20:25:17.159 for 23:15:55
Paths: (2 available, best #2)
 Advertised to update-groups (with more than one peer):
 0.2
 Path #1: Received by speaker 0
 Not advertised to any peer
 Local
 160.0.0.6 (metric 20) from 160.0.0.1 (160.0.0.6)
 Received Label 2
 Origin incomplete, metric 0, localpref 100, valid, internal, labeled-unicast
 Received Path ID 0, Local Path ID 0, version 0
 mac: 10:11:04:64:f2:7f
 Originator: 160.0.0.6, Cluster list: 100.0.0.4
 Path #2: Received by speaker 0
 Advertised to update-groups (with more than one peer):
```

```
 0.2
 Local
 56:78:100::2 from :: (160.0.0.5)
 Origin incomplete, metric 0, localpref 100, weight 32768, valid, redistributed, best,
group-best
 Received Path ID 0, Local Path ID 0, version 9
 mac: 10:11:04:64:f2:7f

/* Verify Ethernet Segments are peering for Dual homing */

Router# show evpn ethernet-segment int bundle-Ether 1

Ethernet Segment Id Interface Nexthops
------------------------ ---------------------------------- --------------------
0056.5656.5656.5656.5601 BE1 160.5.5.5
                              160.6.6.6
-----------------------------------------------------------

/* Verify DF election */

Router# show evpn ethernet-segment int bundle-Ether 1 carving detail
Legend:
 A - Load-balancing mode and Access Protection incompatible,
 B - No Forwarders EVPN-enabled,
 C - Backbone Source MAC missing (PBB-EVPN),
 RT - ES-Import Route Target missing,
 E - ESI missing,
 H - Interface handle missing,
 I - Name (Interface or Virtual Access) missing,
 M - Interface in Down state,
 O - BGP End of Download missing,
 P - Interface already Access Protected,
 Pf - Interface forced single-homed,
 R - BGP RID not received,
 S - Interface in redundancy standby state,
 X - ESI-extracted MAC Conflict
 SHG - No local split-horizon-group label allocated

Ethernet Segment Id Interface Nexthops
------------------------ ---------------------------------- --------------------
0056.5656.5656.5656.5601 BE1 160.5.5.5
 160.6.6.6
 ES to BGP Gates : Ready
 ES to L2FIB Gates : Ready
 Main port :
 Interface name : Bundle-Ether1
 Interface MAC : 008a.9644.acdd
 IfHandle : 0x080004dc
 State : Up
 Redundancy : Not Defined
 ESI type : 0
 Value : 56.5656.5656.5656.5601
 ES Import RT : 5656.5656.5656 (from ESI)
 Source MAC : 0000.0000.0000 (N/A)
 Topology :
 Operational : MH
 Configured : All-active (AApF) (default)
 Primary Services : Auto-selection
 Secondary Services: Auto-selection
 Service Carving Results:
 Forwarders : 161
 Permanent : 10
 EVI:ETag P : 700:1, 701:1, 702:1, 703:1, 704:1, 705:1
 EVI:ETag P : 706:1, 707:1, 708:1, 709:1
```

```
Elected : 76
EVI E : 100, 102, 104, 106, 108, 110
EVI E : 112, 114, 116, 118, 120, 122,
EVI E : 124, 126, 128, 130, 132, 134,
EVI E : 136, 138, 140, 142, 144, 146,
EVI E : 148, 150, 152, 154, 156, 158,
EVI E : 160, 162, 164, 166, 168, 170,
EVI E : 172, 174, 176, 178, 180, 182,
EVI E : 184, 186, 188, 190, 192, 194,
EVI E : 196, 198, 200, 202, 204, 206,
EVI E : 208, 210, 212, 214, 216, 218,
EVI E : 220, 222, 224, 226, 228, 230,
EVI E : 232, 234, 236, 238, 240, 242,
EVI E : 244, 246, 248, 250
Not Elected : 75
EVI NE : 101, 103, 105, 107, 109, 111
EVI NE : 113, 115, 117, 119, 121, 123,
EVI NE : 125, 127, 129, 131, 133, 135,
EVI NE : 137, 139, 141, 143, 145, 147,
EVI NE : 149, 151, 153, 155, 157, 159,
EVI NE : 161, 163, 165, 167, 169, 171,
EVI NE : 173, 175, 177, 179, 181, 183,
EVI NE : 185, 187, 189, 191, 193, 195,
EVI NE : 197, 199, 201, 203, 205, 207,
EVI NE : 209, 211, 213, 215, 217, 219,
EVI NE : 221, 223, 225, 227, 229, 231,
EVI NE : 233, 235, 237, 239, 241, 243,
EVI NE : 245, 247, 249
MAC Flushing mode : STP-TCN
Peering timer : 3 sec [not running]
Recovery timer : 30 sec [not running]
Carving timer : 0 sec [not running]
Local SHG label : 68663
Remote SHG labels : 1
68670 : nexthop 160.6.6.6
```

EVPN IRB: DHCPv4 and DHCPv6 Relay

EVPN IRB: DHCPv4 and DHCPv6 Relay feature provides DHCP support for the end users in EVPN multi-homing Active-Active (MH-AA) deployment scenario. This feature enables reduction of traffic flooding, increase in load sharing at VTEP, faster convergence during link and device failures, and simplification of data center automation.

DHCPv4 and DHCPv6 Relay agents relay request packets, coming over the access interface, to external DHCPv4 and DHCPv4 server to request allocation of addresses (/32) and IANA (::/128) for the end user.

DHCPv4 and DHCPv6 Relay profiles are configured on BVI interfaces which relay DHCPv4 or DHCPv4 requests from Layer 2 (L2) attachment circuit (AC) to external DHCP servers for host IPv4 addresses (/32) and IANA (::128) IPv6 addresses.

This feature is compliant with RFC-6607.

Multi-homing Active-Active EVPN Gateways

Multi-homing Active-Active EVPN Gateways are configured with anycast IP address and MAC addresses. ASR 9000 devices have centralized L2//Layer 3 (L3) gateway. Based on native EVPN and MAC learning, IRB uses distributed anycast IP and anycast MAC address. Static clients are configured with anycast gateway address as the default gateway. DHCP client sends DHCP requests for IP addresses with BVI as the gateway. L2 access can be either single homing or multi-homing, Not all access protocols is supported with IRB. There may or may not be L2 stretch between DC centers. Internet gateway is also included for clients to access external network. No EVPN is configured on the Internet gateway.

EVPN IRB Route Distribution

In EVPN IRB DHCPv4 and DHCPv6, DHCP application processes and DHCP packet forwarding are independent of EVPN IRB L2 and L3 routing. There is no subscriber routing information with the stateless DHCP relay. But DHCP clients work similar to static clients in the EVPN core for L2 and L3 bridging and routing. When the **relay information option**, **relay information option vpn**, **relay information option von-mode cisco** and **relay information option von-mode rfc** commands are configured on the DHCP relay agent, the DHCP relay agent inserts the sub options of DHCP Option 82, such as subnet selection and VPN ID options. These options are considered by DHCP server while allocating IP address.

DHCP clients use the L2 AC interface to access EVPN bridge domain and use BVI interface as default gateway. So the clients must get the IP addresses from the DHCP server as in the same subnet of BVI interface.

DHCP Request Forwarding Path

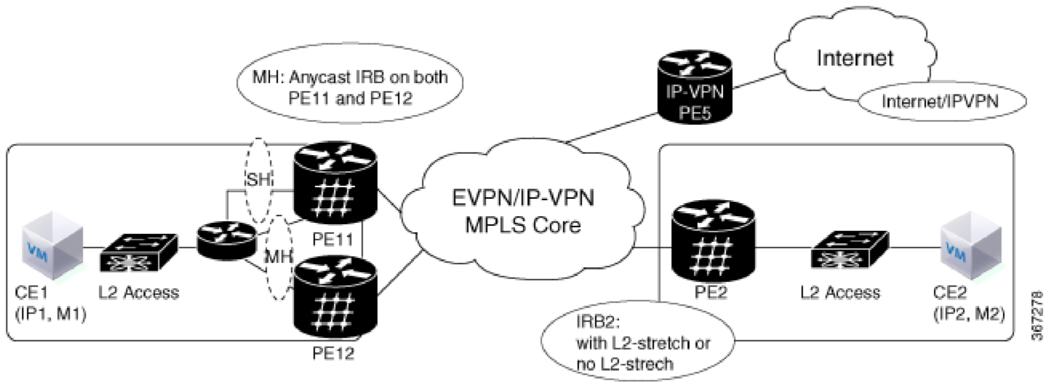
Clients broadcast requests to the access switch with DHAA to EVPN PE routers. The access switch does load balancing. The load balancing configurations in access switch will impact PE in DH-AA and DHCP to send the DHCP requests. The DHCP request reaches the Bridge Domain (BD) BVI interface which is configured with DHCP relay. Because AA PE routers are configured the same IP addresses, BVI IP addresses cannot be used as DHCP relay source IP address.

For DHCPv4, configuring GIADDR field for each DHCP relay profile is allowed. Loopback interface with unique IP addresses can be configured in VRF which is reachable to DHCP servers. Configuring DHCP relay source address is not supported.

In case of DHCPv6 servers, DHCPv6 relay picks up an available Loopback interface IPv6 address as DHCPv6 relay source IP address. After the DHCP clients get the IP address. DHCP clients are not normally routable to DHCP servers. DHCP clients send unicast DHCP renew messages to the DHCP server. If the DHCP servers are not routable, the DHCP unicast messages fail, then the DHCP client sends broadcast rebinding messages with the corrsponding DHCP relay.

In the below figure, DHCP clients are configured on PE11 and PE12.

Figure 101: EVPN IRB with ASR 9000 as Centralized DCI Gateway



Configure EVPN IRB: DHCPv4 and DHCPv6 Relay

Perform the following tasks to configure the EVPN IRB: DHCPv4 and DHCPv6 Relay feature:

```
/* PE11 configuration */

Router(config)# dhcp ipv4
Router(config-dhcpv4)# profile DHCPv4_RELAY relay
Router(config-dhcpv4-relay-profile)# helper-address vrf default 10.20.20.20 giaddr 192.0.2.1
Router(config-dhcpv4-relay-profile)# relay information option vpn
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode rfc
Router(config-dhcpv4-relay-profile)# exit
Router(config-dhcpv4)# exit
Router(config)# interface BVI1 relay profile DHCPv4_RELAY


Router(config)# dhcp ipv6
Router(config-dhcpv6)# profile DHCPv6_RELAY relay
Router(config-dhcpv6-relay-profile)# helper-address vrf default 20::20
Router(config-dhcpv6-relay-profile)# exit
Router(config-dhcpv6)# exit
Router(config)# interface BVI1 relay profile  DHCPv6_RELAY
Router(config-if)# exit
Router(config)# interface Loopback 5
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.255
Router(config-if)# exit
Router(config)# interface BVI1
Router(config-if)# host-routing
Router(config-if)# ipv4 address 10.10.10.2 255.255.255.0
Router(config-if)# ipv6 address 2001:DB8:0:ABCD::1/64
Router(config-if)# ipv6 enable
Router(config-if)# mac-address 0.12.3456

/* PE12 configuration */
Router# dhcp ipv4
Router(config-dhcpv4)# profile DHCPv4_RELAY relay
Router(config-dhcpv4-relay-profile)# helper-address vrf default 10.20.20.20 giaddr 127.0.0.1
Router(config-dhcpv4-relay-profile)# relay information option vpn
Router(config-dhcpv4-relay-profile)# relay information option vpn-mode cisco
Router(config-dhcpv4-relay-profile)# exit
Router(config-dhcpv4)# exit
Router(config)# interface BVI1 relay profile DHCPv4_RELAY
Router(config)#  dhcp ipv6
Router(config-dhcpv6)# profile DHCPv6_RELAY relay
Router(config-dhcpv6-relay-profile)# helper-address vrf default 20::20
Router(config-dhcpv6-relay-profile)# exit
Router(config-dhcpv6)# exit
Router(config) interface BVI1 relay profile DHCPv6_RELAY
Router(config)# interface Loopback 6
Router(config)# exit
Router(config-if)# ipv4 address 127.0.0.1 255.255.255.255

Router(config)# interface BVI1
Router(config-if)# host-routing
Router(config-if)# vrf evpn 1
Router(config-if)# exit
Router(config-if)# ipv4 address 10.10.10.2 255.255.255.0
Router(config-if)# proxy-arp
Router(config-if)# ipv6 address 3000:0:0:8003::2/64
Router(config-if)# ipv6 enable
Router(config-if)# mac-address 1122.3344.5566
```

Running Configuration

```
/* PE11 Configuration */

dhcp ipv4
profile DHCPv4_RELAY relay
  helper-address vrf default 10.20.20.20 giaddr 192.0.2.1
  relay information option vpn
relay information option vpn-mode cisco
!
interface BVI1 relay profile DHCPv4_RELAY
!
dhcp ipv6
profile DHCPv6_RELAY relay
  helper-address vrf default 20::20
!
interface BVI1 relay profile  DHCPv6_RELAY
!
interface Loopback5
ipv4 address 192.0.2.1 255.255.255.0
!
interface BVI1
host-routing
ipv4 address 10.10.10.2 255.255.255.0
ipv6 address  2001:DB8:0:ABCD::1/64
ipv6 enable
mac-address 0.12.3456
!

/* PE12 Configuration */

dhcp ipv4
profile DHCPv4_RELAY relay
  helper-address vrf default 10.20.20.20 giaddr 127.0.0.1
  relay information option vpn
  relay information option vpn-mode cisco
!
interface BVI1 relay profile DHCPv4_RELAY
!
dhcp ipv6
profile DHCPv6_RELAY relay
  helper-address vrf default 20::20
!
interface BVI1 relay profile DHCPv6_RELAY
!
interface Loopback6
ipv4 address 127.0.0.1 255.255.255.255
!
interface BVI1
host-routing
vrf evpn1
ipv4 address 10.10.10.2 255.255.255.0
proxy-arp
ipv6 address 3000:0:0:8003::2/64
ipv6 enable
mac-address 1122.3344.5566
!
```

Verification

Verify the DHCPv4 configuration.

```
Router# show running-configuration dhcp ipv4
Thu Feb 15 21:44:31.550 IST
dhcp ipv4
 profile DHCPv4_RELAY relay
  helper-address vrf default 10.20.20.20 giaddr 192.0.2.1
  relay information option vpn
  relay information option vpn-mode rfc
!
interface BVI1 relay profile DHCPv4_RELAY
!
```

Verify the DHCPv4 relay profile details.

```
Router# show dhcp ipv4 relay profile name DHCPv4_RELAY
Thu Feb 15 21:47:32.247 IST
Profile: DHCPv4_RELAY
Helper Addresses:
        10.20.20.20, vrf default, giaddr 192.0.2.1
Information Option: Disabled
Information Option Allow Untrusted: Disabled
Information Option VPN: Enabled
Information Option VPN Mode: RFC
Information Option Policy: Replace
Information Option Check: Disabled
GIADDR Policy: Keep
Broadcast-flag Policy: Ignore
Mac Mismatch Action: Forward
VRF References:
Interface References: BVI1
```

Verify the DHCPv4 relay packet statistics.

```
Router# show dhcp ipv4 relay statistics

Fri Feb 16 12:34:51.202 IST

      VRF                     |     RX   |    TX |  DR |
-------------------------------------------------------
default                       |     4    |    4  |  0  |
**nVSatellite                 |     0    |    0  |  0  |
```

Verify DHCPv4 relay packet statistics in detail.

```
Router# show dhcp vrf default ipv4 relay statistics
Fri Feb 16 12:36:05.544 IST

DHCP IPv4 Relay Statistics for VRF default:

     TYPE          |    RECEIVE    |    TRANSMIT   |     DROP      |
-------------------------------------------------------------------
DISCOVER          |            1  |            1  |            0  |
OFFER             |            1  |            1  |            0  |
REQUEST           |            1  |            1  |            0  |
DECLINE          |            0  |            0  |            0  |
ACK              |            1  |            1  |            0  |
NAK              |            0  |            0  |            0  |
RELEASE          |            0  |            0  |            0  |
INFORM           |            0  |            0  |            0  |
LEASEQUERY       |            0  |            0  |            0  |
LEASEUNASSIGNED  |            0  |            0  |            0  |
LEASEUNKNOWN     |            0  |            0  |            0  |
LEASEACTIVE      |            0  |            0  |            0  |
BOOTP-REQUEST    |            0  |            0  |            0  |
BOOTP-REPLY      |            0  |            0  |            0  |
BOOTP-INVALID    |            0  |            0  |            0  |
```

Verify the DHCPv6 configuration.

```
Router# **show running-configuration dhcp ipv6**
Fri Feb 16 15:40:52.721 IST
dhcp ipv6
profile DHCPv6_RELAY relay
  helper-address vrf default 20::20
!
interface BVI1 relay profile DHCPv6_RELAY
!
```

Verify DHCPv6 relay profile.

```
Router# **show dhcp ipv6 relay statistics**
Fri Feb 16 15:41:00.456 IST

      VRF                      |     RX   |    TX  |  DR |
------------------------------------------------------------------------------------------
default                        |     4    |    4   |  0  |
**nVSatellite                  |     0    |    0   |  0  |
```

Verify DHCPv6 relay packet statistics in detail.

```
Routerr# **show dhcp ipv6 relay statistics vrf default**
Fri Feb 16 15:41:09.991 IST

DHCP IPv6 Relay Statistics for VRF default:

     TYPE         |    RECEIVE    |    TRANSMIT   |     DROP      |
-------------------------------------------------------------------
SOLICIT          |            1  |            0  |            0  |
ADVERTISE        |            0  |            1  |            0  |
REQUEST          |            1  |            0  |            0  |
REPLY            |            0  |            1  |            0  |
CONFIRM          |            0  |            0  |            0  |
DECLINE          |            0  |            0  |            0  |
RENEW            |            0  |            0  |            0  |
REBIND           |            0  |            0  |            0  |
RELEASE          |            0  |            0  |            0  |
RECONFIG         |            0  |            0  |            0  |
INFORM           |            0  |            0  |            0  |
RELAY_FWD        |            0  |            0  |            0  |
RELAY_REP          |              0 |              0  |              0  |
LEASEQUERY         |              0  |             0  |              0  |
LEASEQUERY_REP     |              0  |             0  |              0  |
LEASEQUERY_DONE    |              0  |             0  |              0  |
LEASEQUERY_DATA    |              0  |             0  |              0  |
```

Duplicate IP Address Detection

The Duplicate IP Address Detection feature automatically detects any host with a duplicate IP address and blocks all MAC-IP routes that have a duplicate IP address.

This protects the network from hosts that are assigned duplicate IP addresses unintentionally or by malicious intent in an EVPN fabric. Hosts with duplicate IP address cause unnecessary churn in a network and causes traffic loss to either or both the hosts with the same IP address.

The system handles mobility of EVPN hosts by keeping track of MAC and IP addresses as they move from one host to another. If two hosts are assigned the same IP address, the IOS XR system keeps learning and re-learning MAC-IP routes from both the hosts. Each time it learns the MAC-IP route from one host, it is counted as one move since the newly learnt route supersedes the route previously learnt from the other host. This continues back and forth until the IP address is marked as duplicate based on the configured parameters.

It uses the following parameters to determine when an IP address should be marked as duplicate, and frozen or unfrozen as it moves between different hosts. The configurable parameters are:

- **move-interval**: The period within which a MAC or IP address has to move certain number of times between different hosts to be considered as duplicate and frozen temporarily. This number is specified in the **move-count** parameter.
- **move-count**: The number of times a MAC or IP address has to move within the interval specified for the **move-interval** parameter between different hosts to be considered a duplicate.
- **freeze-time**: The length of time a MAC or IP address is locked after it has been detected as a duplicate. After this period, the IP address is unlocked and it is allowed to learn again.
- **retry-count**: The number of times a MAC or IP address is unlocked after it has been detected as a duplicate before it is frozen permanently.

The system maintains a count of the number of times an IP address has been moved from one host to another host, either to another local host or to a host behind a remote Top of Rack (TOR). If an IP address moves certain number of times specified in the **move-count** parameter within the interval specified in the **move-interval** parameter is considered a duplicate IP address. All MAC-IP routes with that IP address is frozen for the time specified in the **freeze-time** parameter. A syslog notifies the user that the particular IP address is frozen. While an IP address is frozen, any new MAC-IP routes or updates to existing MAC-IP routes with the frozen IP address are ignored.

After **freeze-time** has elapsed, the corresponding MAC-IP routes are unfrozen and the value of the **move-count** is reset to zero. For any unfrozen local MAC-IP routes, an ARP probe and flush are initiated while the remote MAC-IP routes are put in the probe mode. This restarts the duplicate detection process.

The system also maintains the information about the number of times a particular IP address has been frozen and unfrozen. If an IP address is marked as duplicate after it is unfrozen **retry-count** times, it is frozen permanently until user manually unfreezes it. Use the following commands to manually unfreeze frozen MAC, IPv4 and IPV6 addresses respectively:

- **clear l2route evpn mac** { *mac-address* } | **all** [**evi** *evi*] **frozen-flag**

- **clear l2route evpn ipv4**{*ipv4-address*} | **all** [**evi** *evi*] **frozen-flag**
- **clear l2route evpn ipv6**{*ipv6-address*} | **all** [**evi** *evi*] **frozen-flag**

Configure Duplicate IP Address Detection

Perfrom these tasks to configure Duplicate IP Address Detection feature.

Configuration Example

```
/* Ipv4 Address Duplicate Detection Configuration */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# host ipv4-address duplicate-detection
RP/0/RSP0/CPU0:router(config-evpn-host-ipv4-addr)# move-count 2
RP/0/RSP0/CPU0:router(config-evpn-host-ipv4-addr)# freeze-time 10
RP/0/RSP0/CPU0:router(config-evpn-host-ipv4-addr)# retry-count 2
RP/0/RSP0/CPU0:router(config-evpn-host-ipv4-addr)# commit

/* Ipv6 Address Duplicate Detection Configuration */

RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# evpn
RP/0/RSP0/CPU0:router(config-evpn)# host ipv6-address duplicate-detection
RP/0/RSP0/CPU0:router(config-evpn-host-ipv6-addr)# move-count 2
RP/0/RSP0/CPU0:router(config-evpn-host-ipv6-addr)# freeze-time 10
RP/0/RSP0/CPU0:router(config-evpn-host-ipv6-addr)# retry-count 2
RP/0/RSP0/CPU0:router(config-evpn-host-ipv6-addr)# commit
```

Running Configuration

This section shows the running configuration to detect duplicate IP address.

```
evpn
 host ipv4-address duplicate-detection
  move-count 2
  freeze-time 10
  retry-count 2
 !
evpn
 host ipv6-address duplicate-detection
  move-count 2
  freeze-time 10
  retry-count 2
 !
```

Verification

The show output given in the following section display the details of the duplicate IP address detection and recovery parameters.

```
RP/0/RSP0/CPU0:router#show l2route evpn mac-ip all detail

Flags:  (Stt)=Static; (L)=Local; (R)=Remote; (F)=Flood;
        (N)=No Redistribution; (Rtr)=RP/0/RSP0/CPU0:router MAC; (B)=Best Route;
        (S)=Peer Sync; (Spl)=Split; (Rcv)=Recd;
        (D)=Duplicate MAC; (Z)=Frozen MAC;

Topo ID    Mac Address     IP Address  Prod   Next Hop(s)        Seq No  Flags
Opaque Data Type    Opaque Data Len   Opaque Data Value
-------    -----------     ----------  ----   ----------         ------  -----
----------------    ---------------   -----------------
33         0022.6730.0001  10.130.0.2  L2VPN  Bundle-Ether6.1300  0      SB 0 12
0x06000000
```

Related Topics

- Duplicate IP Address Detection, on page 736

Associated Commands

- evpn host ipv4-address duplicate-detection
- evpn host ipv6-address duplicate-detection
- show l2route evpn mac-ip all detail

EVPN E-Tree Using RT Constraints

The EVPN E-Tree using RT constraints feature enables you to configure BGP RT import and export policies for an attachment circuit. This feature allows you to define communication between the leaf and root nodes. The provider edge (PE) nodes can receive L2 traffic either from the attachment circuit (AC) of a bridge domain (BD) or from the remote PE node. For a given BD, L2 communication can only happen from root to leaf and leaf to root. This feature does not allow any L2 communication between the ACs of two or more leafs. This feature uses two BGP RTs for every EVI. Associate one RT with root ACs and the other with leaf ACs. For example, there are two distinct sets of RTs, one for root-rt and another for leaf-rt.

This feature provides you with the following benefits by performing filtering of unicast and multicast traffic at the ingress PE nodes:

- Achieve efficiency of the BGP MAC routes scale
- Reduce the consumption of hardware resources
- Utilize the link bandwidth efficiently

Rules for Import and Export Policies under the BGP of EVPN EVI Instances

- Root PE exports its ROOT-RT using BGP export policy. It also imports other ROOT-RT from the corresponding root PE for the same EVI. This is necessary where there is more than one root for a particular BD and EVPN EVI. For example, in a multihome active-active scenario or multihome port-active and single-active scenarios.
- Root PE imports LEAF-RT using BGP import policy for a EVPN EVI. This enables the root to be aware of all remote L2 MAC addresses through EVPN RT2 advertisement of leaf PE node for a given E-Tree EVI.
- Leaf PE exports its LEAF-RT using BGP export policy to let the root to be aware of the reachability of its directly connected L2 endpoints through EVPN RT2 advertisement.

- Leaf PE imports ROOT-RT using BGP import policy. It helps the leaf to know about the L2 endpoints which are reachable through the AC of BD under EVPN EVI instance of root PE. You must not import LEAF-RT using BGP Import policy to avoid L2 Communication between two leaf PEs.
- Use split-horizon filtering to block traffic among leaf ACs on a BD for a given E-Tree EVI.

The BGP import and export policies applies to all EVPN RTs along with the RT2 advertisement.

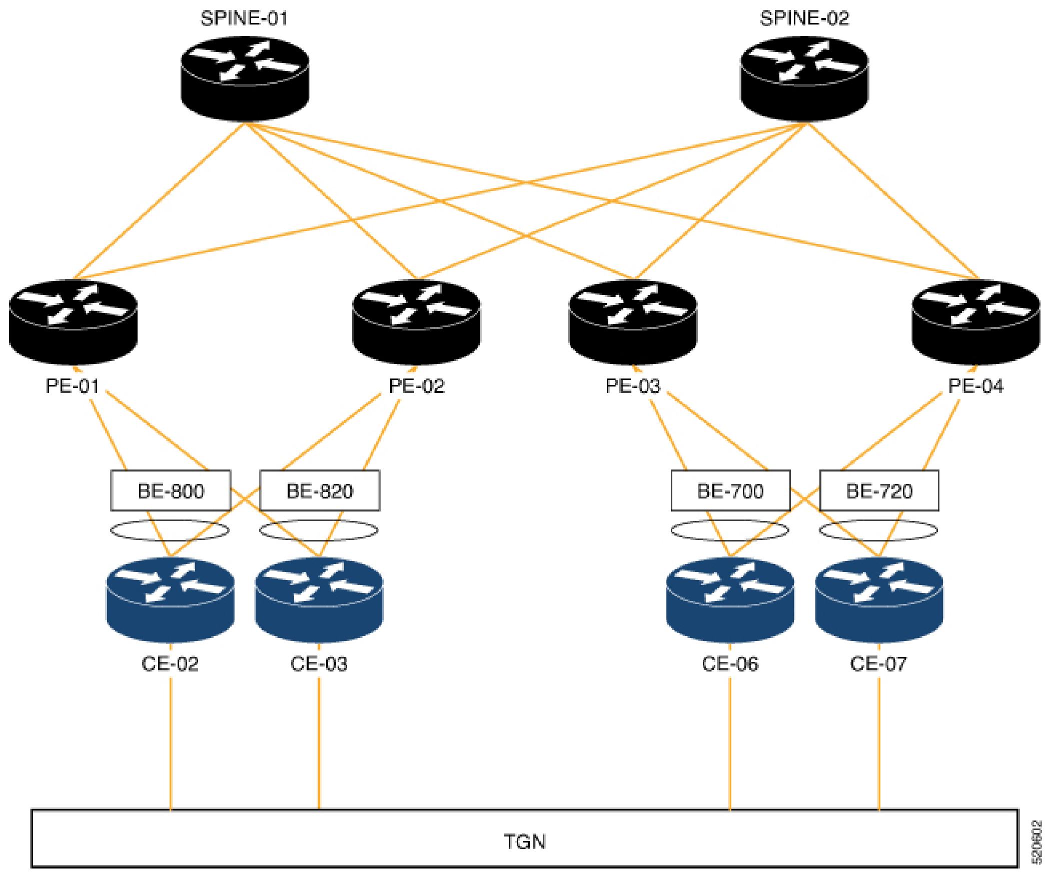
MAC Address Learning

- L2 MAC addresses are learnt on AC of a particular BD on leaf PE as type LOCAL. The same MAC address is advertised to root PE as EVPN RT2. On the remote root PE, the MAC table replicates the entry of MAC address with the learn type as L2VPN. Also, it associates the MPLS label of its BGP peer, which advertises RT2 to root PE node.
- L2 MAC addresses are learnt on AC of a particular BD on the root as type LOCAL. The same MAC address is advertised to peer root (except for MH A/A) or leaf PE as EVPN RT2. On the remote root PE or leaf PE, the MAC table replicates the entry of MAC address with the learn type as L2VPN. Also, it associates the MPLS label of its BGP peer, which advertises RT2 to PE node.
- L2 MAC addresses are learnt on AC of a particular BD on the root as type LOCAL. The same MAC address is advertised to peer root for MH A/A as EVPN RT2. The MAC table of the peer root node synchronizes the replicated entry of MAC address with the learn type as L2VPN for same the ESI and with the same AC as the next hop. This avoids flooding and duplication of known unicast traffic.

The following scenario describes the feature topology:

CE with Multihoming Active-Active and CE with Multihoming Active-Active

Consider a topology where you connect CE-02 and CE-03 to PE-01 and PE-02. Both the CEs are in multihoming active-active mode. Connect CE-02 to PE-01 and PE-02 using AC BE-800.305. Connect CE-03 to PE-01 and PE-02 using AC BE-820.305. Connect CE-06 and CE-07 to PE-03 and PE-04. Connect CE-06 to PE-03 and PE-04 using AC BE-700.305. Connect CE-07 to PE-03 and PE-04 using AC BE-720.305. Associate the bridge domain BD-305 with other AC on the respective PEs along with EVI-305 instance. Configure the respective RT on root and leaf with its import and export RTs for EVI-305. Configure PE-01 and PE-02 as root. Configure PE-03 and PE-04 as leaf.



Configuration

Perform the following tasks on PE-01, PE-02, PE-03, and PE-04.

- Configure bridge domain
- Configure attachment circuit
- Configure EVPN EVI
- Configure bundle Ethernet
- Configure EVPN interface

Note Use the **etree rt-leaf** command only if the leaf sites are in the EVPN all-active multihoming mode and not required for EVPN single homing mode.

Configuration Example

```
/* Configure PE-01 (as root) */

/* Configure bridge domain */
Router # configure
```

```
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether800.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# interface Bundle-Ether820.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether800.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether820.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:305
Router(config-evpn-instance-bgp)# route-target import 1001:5305
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether800
Router(config-if)# lacp system mac 00aa.aabb.2020
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether820
Router(config-if)# lacp system mac 00aa.aabb.2222
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether800
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.00
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether820
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.20
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0020
Router(config-evpn-ac-es)# commit


/* Configure PE-02 (as root) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether800.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# interface Bundle-Ether820.305
Router(config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether800.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether820.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:305
Router(config-evpn-instance-bgp)# route-target import 1001:5305
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether800
Router(config-if)# lacp system mac 00aa.aabb.2020
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether820
Router(config-if)# lacp system mac 00aa.aabb.2222
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit
```

```
/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether800
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.00
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether820
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.88.88.88.88.88.88.88.20
Router(config-evpn-ac-es)# bgp route-target 0001.0000.0020
Router(config-evpn-ac-es)# commit


/* Configure PE-03 (as leaf) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether700.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface Bundle-Ether720.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether700.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether720.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:5305
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# etree
Router(config-evpn-instance-etree)# rt-leaf
Router(config-evpn-instance)# exit
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether700
Router(config-if)# lacp system mac 00aa.aabb.1010
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether720
Router(config-if)# lacp system mac 00aa.aabb.1212
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether700
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.00
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether720
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.20
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0020
Router(config-evpn-ac-es)# commit


/* Configure PE-04 (as leaf) */

/* Configure bridge domain */
Router # configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group EVPN_BD
Router(config-l2vpn-bg)# bridge-domain evpn_bvi_305
Router(config-l2vpn-bg-bd)# interface Bundle-Ether700.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# interface Bundle-Ether720.305
Router(config-l2vpn-bg-bd-ac)# split-horizon group
Router (config-l2vpn-bg-bd-ac)# exit
Router (config-l2vpn-bg-bd)# evi 305
Router (config-l2vpn-bg-bd-evi)# commit

/* Configure attachment circuit */
Router# configure
Router(config)# interface Bundle-Ether700.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

Router# configure
Router(config)# interface Bundle-Ether720.305 l2transport
Router(config-l2vpn-subif)# encapsulation dot1q 305
Router(config-l2vpn-subif)# rewrite ingress tag pop 1 symmetric
Router(config-l2vpn-subif)# commit

/* Configure EVPN EVI */
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 305
Router(config-evpn-instance)# bgp
Router(config-evpn-instance-bgp)# route-target import 1001:305
Router(config-evpn-instance-bgp)# route-target export 1001:5305
Router(config-evpn-instance-bgp)# exit
Router(config-evpn-instance)# etree
Router(config-evpn-instance-etree)# rt-leaf
Router(config-evpn-instance)# exit
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# advertise-mac
Router(config-evpn-instance-mac)# commit

/* Configure bundle Ethernet */
Router# configure
Router(config)# interface Bundle-Ether700
Router(config-if)# lacp system mac 00aa.aabb.1010
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

Router# configure
Router(config)# interface Bundle-Ether720
Router(config-if)# lacp system mac 00aa.aabb.1212
Router(config-if)# lacp switchover suppress-flaps 300
Router(config-if)# lacp cisco enable link-order signaled
Router(config-if)# bundle wait-while 100
Router(config-if)# commit

/* Configure EVPN interface */
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether700
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.00
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0001
Router(config-evpn-ac-es)# commit

Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether720
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.77.77.77.77.77.77.77.20
Router(config-evpn-ac-es)# bgp route-target 0000.0000.0020
Router(config-evpn-ac-es)# commit
```

Running Configuration

This section shows the PE-01, PE-02, PE-3, and PE-04 running configuration.

```
/* PE-01 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether800.305
   !
   interface Bundle-Ether820.305
   !
   evi 305
   !
  !
 !
interface Bundle-Ether800.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
```

```
!
interface Bundle-Ether820.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:305
   route-target import 1001:5305
  !
  control-word-disable
  advertise-mac
  !
 !
!
interface Bundle-Ether800
 lacp system mac 00aa.aabb.2020
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether820
 lacp system mac 00aa.aabb.2222
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether800
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.00
   bgp route-target 0001.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether820
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.20
   bgp route-target 0001.0000.0020
  !
 !
!


/* PE-02 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether800.305
   !
   interface Bundle-Ether820.305
   !
   evi 305
   !
  !
 interface Bundle-Ether800.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether820.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
```

```
!
evpn
 evi 305
  bgp
   route-target import 1001:305
   route-target export 1001:305
   route-target import 1001:5305
  !
  control-word-disable
  advertise-mac
  !
 !
interface Bundle-Ether800
 lacp system mac 00aa.aabb.2020
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether820
 lacp system mac 00aa.aabb.2222
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether800
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.00
   bgp route-target 0001.0000.0001
  !
 !
evpn
 interface Bundle-Ether820
  ethernet-segment
   identifier type 0 00.88.88.88.88.88.88.88.20
   bgp route-target 0001.0000.0020
  !
 !


/* PE-03 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether700.305
    split-horizon group
   !
   interface Bundle-Ether720.305
    split-horizon group
   !
   evi 305
   !
  !
 !
interface Bundle-Ether700.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether720.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
  bgp
```

```
   route-target import 1001:305
   route-target export 1001:5305
  !
  etree
   rt-leaf
  !
  control-word-disable
  advertise-mac
  !
 !
!
interface Bundle-Ether700
 lacp system mac 00aa.aabb.1010
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether720
 lacp system mac 00aa.aabb.1212
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether700
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.00
   bgp route-target 0000.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether720
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.20
   bgp route-target 0000.0000.0020
  !
 !
!


/* PE-04 Configuration */
l2vpn
 bridge group EVPN_BD
  bridge-domain evpn_bvi_305
   interface Bundle-Ether700.305
    split-horizon group
   !
   interface Bundle-Ether720.305
    split-horizon group
   !
   evi 305
   !
  !
 !
interface Bundle-Ether700.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
interface Bundle-Ether720.305 l2transport
 encapsulation dot1q 305
 rewrite ingress tag pop 1 symmetric
!
evpn
 evi 305
```

```
  bgp
   route-target import 1001:305
   route-target export 1001:5305
  !
  etree
   rt-leaf
  !
  control-word-disable
  advertise-mac
  !
 !
!
interface Bundle-Ether700
 lacp system mac 00aa.aabb.1010
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
interface Bundle-Ether720
 lacp system mac 00aa.aabb.1212
 lacp switchover suppress-flaps 300
 lacp cisco enable link-order signaled
 bundle wait-while 100
!
evpn
 interface Bundle-Ether700
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.00
   bgp route-target 0000.0000.0001
  !
 !
!
evpn
 interface Bundle-Ether720
  ethernet-segment
   identifier type 0 00.77.77.77.77.77.77.77.20
   bgp route-target 0000.0000.0020
  !
 !
!
```

Verification

This section shows how the L2 MAC addresses are synchronized as LOCAL and L2VPN with multihoming active-active peers PE. Also, the root PE is aware of MAC addresses learnt on leaf PE remotely through RT2 advertisements.

```
Router:PE-01# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- ----------------------------------------
204      001f.0100.0001 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0001 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0002 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0002 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0003 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0003 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0004 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0004 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0005 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0005 L2VPN       Bundle-Ether820.305, N/A
204      0020.0100.0001 L2VPN       26791/I/ME, N/A
204      0020.0100.0002 L2VPN       26791/I/ME, N/A
```

```
204      0020.0100.0003 L2VPN       26791/I/ME, N/A
204      0020.0100.0004 L2VPN       26791/I/ME, N/A
204      0020.0100.0005 L2VPN       26791/I/ME, N/A
204      0021.0100.0001 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0002 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0003 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0004 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0005 LOCAL       Bundle-Ether800.305, N/A
204      0022.0100.0001 L2VPN       26790/I/ME, N/A
204      0022.0100.0002 L2VPN       26790/I/ME, N/A
204      0022.0100.0003 L2VPN       26790/I/ME, N/A
204      0022.0100.0004 L2VPN       26790/I/ME, N/A
204      0022.0100.0005 L2VPN       26790/I/ME, N/A

Router:PE-02# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- ----------------------------------------
204      001f.0100.0001 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0001 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0002 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0002 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0003 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0003 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0004 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0004 L2VPN       Bundle-Ether820.305, N/A
204      001f.0100.0005 LOCAL       Bundle-Ether820.305, N/A
204      001f.0100.0005 L2VPN       Bundle-Ether820.305, N/A
204      0020.0100.0001 L2VPN       27367/I/ME, N/A
204      0020.0100.0002 L2VPN       27367/I/ME, N/A
204      0020.0100.0003 L2VPN       27367/I/ME, N/A
204      0020.0100.0004 L2VPN       27367/I/ME, N/A
204      0020.0100.0005 L2VPN       27367/I/ME, N/A
204      0021.0100.0001 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0002 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0003 L2VPN       Bundle-Ether800.305, N/A
204      0021.0100.0004 LOCAL       Bundle-Ether800.305, N/A
204      0021.0100.0005 L2VPN       Bundle-Ether800.305, N/A
204      0022.0100.0001 L2VPN       27366/I/ME, N/A
204      0022.0100.0002 L2VPN       27366/I/ME, N/A
204      0022.0100.0003 L2VPN       27366/I/ME, N/A
204      0022.0100.0004 L2VPN       27366/I/ME, N/A
204      0022.0100.0005 L2VPN       27366/I/ME, N/A
```

The following output shows how the multihoming PE is aware of its local L2 MAC addresses as well as the MAC addresses learnt on the root node only. Leaf multihoming PE is not aware of any other MAC addresses learnt on other leaf PE nodes except if they are learnt on a multihoming active-active ethernet-segment on the peer leaf PE.

```
Router:PE-03# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- ----------------------------------------
200      0011.0100.0003 L2VPN       30579/I/ME, N/A
200      0011.0100.0005 L2VPN       30579/I/ME, N/A
204      001f.0100.0001 L2VPN       30588/I/ME, N/A
204      001f.0100.0002 L2VPN       30588/I/ME, N/A
204      001f.0100.0003 L2VPN       30588/I/ME, N/A
204      001f.0100.0004 L2VPN       30588/I/ME, N/A
204      001f.0100.0005 L2VPN       30588/I/ME, N/A
204      0020.0100.0001 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0001 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0002 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0002 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0003 LOCAL       Bundle-Ether720.305, N/A
```

```
204      0020.0100.0003 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0004 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0004 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0005 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0005 L2VPN       Bundle-Ether720.305, N/A
204      0021.0100.0001 L2VPN       30587/I/ME, N/A
204      0021.0100.0002 L2VPN       30587/I/ME, N/A
204      0021.0100.0003 L2VPN       30587/I/ME, N/A
204      0021.0100.0004 L2VPN       30587/I/ME, N/A
204      0021.0100.0005 L2VPN       30587/I/ME, N/A
204      0022.0100.0001 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0001 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0002 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0002 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0003 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0003 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0004 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0004 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0005 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0005 L2VPN       Bundle-Ether700.305, N/A

Router:PE-04# show l2route evpn mac all
Topo ID  Mac Address    Producer    Next Hop(s)
-------- -------------- ----------- ----------------------------------------
200      0011.0100.0003 L2VPN       30545/I/ME, N/A
200      0011.0100.0005 L2VPN       30545/I/ME, N/A
204      001f.0100.0001 L2VPN       30550/I/ME, N/A
204      001f.0100.0002 L2VPN       30550/I/ME, N/A
204      001f.0100.0003 L2VPN       30550/I/ME, N/A
204      001f.0100.0004 L2VPN       30550/I/ME, N/A
204      001f.0100.0005 L2VPN       30550/I/ME, N/A
204      0020.0100.0001 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0001 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0002 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0002 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0003 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0003 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0004 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0004 L2VPN       Bundle-Ether720.305, N/A
204      0020.0100.0005 LOCAL       Bundle-Ether720.305, N/A
204      0020.0100.0005 L2VPN       Bundle-Ether720.305, N/A
204      0021.0100.0001 L2VPN       30549/I/ME, N/A
204      0021.0100.0002 L2VPN       30549/I/ME, N/A
204      0021.0100.0003 L2VPN       30549/I/ME, N/A
204      0021.0100.0004 L2VPN       30549/I/ME, N/A
204      0021.0100.0005 L2VPN       30549/I/ME, N/A
204      0022.0100.0001 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0001 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0002 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0002 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0003 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0003 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0004 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0004 L2VPN       Bundle-Ether700.305, N/A
204      0022.0100.0005 LOCAL       Bundle-Ether700.305, N/A
204      0022.0100.0005 L2VPN       Bundle-Ether700.305, N/A
```

Related Topics

- EVPN E-Tree Using RT Constraints, on page 738

Associated Commands

- etree rt-leaf
- show l2route evpn mac all

EVPN E-Tree Per-PE (Scenario1b)

Table 25: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-Tree Per-PE (Scenario1b)	Release 7.5.1	This feature allows you to configure an attachment circuit on a PE device either as a root site or a leaf site using the **etree leaf** label for a given bridge-domain. By preventing communication among leaf ACs connected to the same PE and belonging to the same bridge-domain, you can segregate traffic received and sent from different geographical locations. This segregation helps in securing traffic and Denial of Service Prevention (DoS). This feature is compliant with RFC 8317.

EVPN Ethernet Tree (E-Tree) is a rooted-multipoint Ethernet service over MPLS core and enables you to define attachment circuits (ACs) as either a root site or a leaf site. The provider edge (PE) nodes can receive L2 traffic either from the attachment circuit (AC) of a bridge domain (BD) or from the remote PE node. For a given BD, L2 communication can only happen from root to leaf and leaf to root, and root to root. L2 communication between the ACs of two or more leafs is not allowed.

You can implement E-Tree in the following two ways:

- Scenario 1 - All ACs at a particular PE for a given BD can be either root or leaf site and all traffic for an EVI from a PE in the network is from either a root or a leaf. In this scenario you have two options to configure E-Tree:
  - Scenario 1a - You can configure E-Tree with route-targets (RT) constraints using two RTs per EVI. For more information, see the *EVPN E-Tree Using RT Constraints* section.
  - Scenario 1b - You can configure E-Tree without route-targets (RT) constraints and using **etree leaf** label.
- Scenario 2 - A PE for a given EVI can have both root and leaf sites. The root and leaf designation increases from bridge-domain level in Scenario 1, to a per AC level. For more information, see the *EVPN ETREE Per-AC (Scenario 2)* section.

Scenario 1b

In this scenario, you can configure E-Tree without route-targets (RT) constraints and using **etree leaf** label.

When you configure E-Tree with Scenario 1b, for known unicast traffic, MAC advertisements originating from a leaf site is identified with an **etree leaf** label to classify that the source is a leaf. Ingress filtering is performed, and traffic originating at leaf AC destined for a remote leaf MAC is dropped. If the remote PE is also a leaf, the ingress traffic from the source leaf is dropped. If the remote PE is a root, the ingress traffic from the source leaf is forwarded.

For BUM traffic, egress filtering is performed and leaf nodes transmit an **etree leaf** label to identify that leaf sites are connected to the PE. Then, at the ingress node, BUM traffic originating from a leaf node is tagged with the corresponding remote **etree leaf** label. At the egress PE, traffic is tagged with the matching **etree leaf** label that is dropped at leaf ACs.

Scenario 1b Behavior

- E-Tree leaf is configured per bridge domain. If there is no leaf configuration, the bridge domain is always a root.
- All ACs inherit E-Tree leaf designation from the bridge domain.
- Split-horizon group between ACs of a leaf is enabled automatically.
- All local MACs learned under the BD is advertised to BGP with **etree leaf** indicator.
- Upon first leaf configuration, a special E-Tree ethernet segment with ESI-0 is created to allocate a split-horizon label, referred to as the local etree leaf label.
- ES/EAD with ESI-0 (ES-0/EAD) is advertised to BGP with etree leaf label.

You can use scenario 1b to interact with scenario 2 when a remote node is NCS 5500, NCS 540, or NCS 560, and configure E-Tree using scenario 1b with ASR 9000.

Configure EVPN E-Tree Per-PE (Scenario1b)

Perform this task to configure EVPN E-Tree Per-PE (Scenario1b).

Configure EVPN E-Tree leaf per bridge domain.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd_1
Router(config-l2vpn-bg-bd)# etree
Router(config-l2vpn-bg-bd-etree)# leaf
Router(config-l2vpn-bg-bd-etree)# interface Bundle-Ether400.1
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd-etree)#  interface Bundle-Ether401.1001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd-etree)# interface Bundle-Ether4701.2001
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd-ac)# exit
Router(config-l2vpn-bg-bd)# routed interface BVI1
Router(config-l2vpn-bg-bd-bvi)# split-horizon group core
Router(config-l2vpn-bg-bd-bvi)# exit
Router(config-l2vpn-bg-bd)# evi 200
Router(config-l2vpn-bg-bd-evi)# commit
```

Running Configuration

```
l2vpn
 bridge group bg1
  bridge-domain bd_1
   etree
```

```
      leaf
     !
     interface Bundle-Ether400.1
     !
     interface Bundle-Ether401.1001
     !
     interface Bundle-Ether4701.2001
     !
     routed interface BVI1
      split-horizon group core
     !
     evi 200
     !
   !
  !
```

EVPN ETREE Per-AC (Scenario 2)

Table 26: Feature History Table

Feature Name	Release Information	Feature Description
EVPN E-TREE Per-AC (Scenario 2)	Release 7.5.1	This feature allows a provider edge (PE) device to have both root and leaf sites for a given EVI. This feature increases the granularity of leaf designation from the entire bridge to AC bridge ports; ACs under a bridge may be root or leaf.

Customer sites represented by ACs can be defined as root or leaf. PE can receive traffic from both root and leaf ACs for a given EVI from a remote node. An EVI can be associated with both root(s) and leaf(s). If an AC is not configured as E-Tree leaf, it is considered as root by default.

In Scenario 2, a PE for a given EVI can have both root and leaf sites. The granularity of root or leaf designation increases from bridge-domain level in Scenario 1 to a per AC level. Traffic for an EVI from a PE in the network can be from either a root or a leaf site.

Scenario 2 Behavior

For unicast traffic:

- Remote PE performs ingress filtering to prevent traffic from being sent unnecessarily over the core to be filtered at egress PE.
- PE indicates if the MAC address is associated with a root or a leaf.
- MAC advertisements originating from a leaf site are colored with a leaf-indication flag in an extended community; routes that do not have this flag are from a root site.
- Remote PEs perform ingress filtering, when MAC is programmed and the leaf-indication is present, a cross-check is performed with the originating AC; if the AC is also a leaf, packets will not be forwarded.
- Supports E-Tree extcomm of type 0x06 (EVPN) and sub-type 0x05 used for leaf-indication for known-unicast and BUM traffic.
- Enable unknown unicast suppression at EVIs connected to both root and leaf sites to prevent egress unknown unicast traffic arriving at an EVI from being flooded to ACs. This eliminates the leaf-to-leaf traffic during a BD MAC flush.

- MAC advertises local ESI and there is no leaf indicator from root.

  When processing the root sync-route, the root or leaf status of the individual AC is considered, instead of the entire bridge-domain. If a root MAC with a matching local ESI is received, and if the corresponding AC is a leaf, a syslog message is generated for the misconfiguration.

For BUM traffic:

- The PE performs egress filtering for BUM traffic. BUM traffic originating from leaf sites is filtered at egress nodes if the destination is also a leaf.
- A PE with leaf sites allocate a leaf label, and communicate this label to remote PEs using an ES/EAD route with ESI 0 with the ETREE extended community.
- BUM traffic originating from single-homed leaf AC is tagged with destination **etree leaf** label.
- BUM traffic originating from single-homed root AC is not tagged with any ESI or **etree leaf** label.
- BUM traffic originating from multi-homed leaf AC is tagged with destination **etree leaf** label.
- BUM traffic originating from multi-homed root AC is tagged with ESI label.
- The ingress PE tags MPLS frames with the **etree leaf** label for traffic originating from a leaf site; this label allows the disposition PE to perform egress filtering to native EVPN ESI label filtering.
- Intra-PE forwarding between leaf sites is prevented by putting all leaf ACs under a given bridge domain in a single split-horizon group.

Restrictions

- This feature is supporeted only on physical and bundle interfaces.
- You can use either Scenario 1 or Scenario 2 for a given EVI within a network.
- Scenario 2 interoperates only with scenario 1b and vice-versa.
- You can use scenario 1b to interact with scenario 2 when a remote node is NCS 5500, NCS 540, or NCS 560, and configure E-Tree using scenario 1b with ASR 9000.

CHAPTER 13

EVPN Virtual Private Wire Service (VPWS)

The EVPN-VPWS is a BGP control plane solution for point-to-point services. It implements the signaling and encapsulation techniques for establishing an EVPN instance between a pair of PEs. It has the ability to forward traffic from one network to another without MAC lookup. The use of EVPN for VPWS eliminates the need for signaling single-segment and multi-segment PWs for point-to-point Ethernet services. The EVPN-VPWS technology works on IP and MPLS core; IP core to support BGP and MPLS core for switching packets between the endpoints.

EVPN-VPWS support both single-homing and multi-homing.

Note When both MPLS and SRv6 are configured in the core, EVPN VPWS services cannot co-exist with SRv6 and MPLS.

Supported Modes for EVPN-VPWS

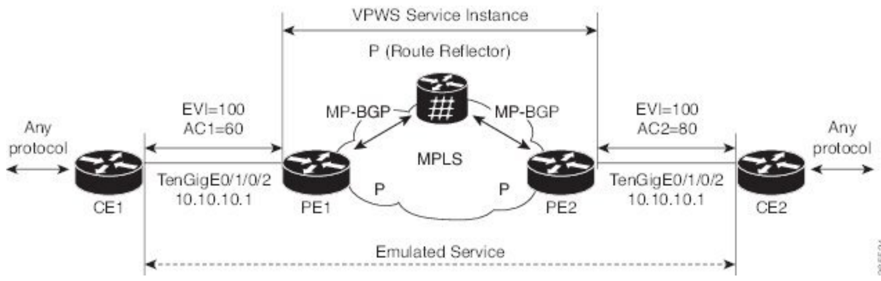
EVPN-VPWS supports the following modes:

- Single-homed - Enables you to connect a customer edge (CE) device to one provider edge (PE) device.
- Multi-homed - Enables you to connect a customer edge (CE) device to more than one provider edge (PE) device. Multihoming ensures redundant connectivity. The redundant PE device ensures that there is no traffic disruption when there is a network failure. Following are the types of multihoming:
  - Single-Active - In single-active mode only a single PE among a group of PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.
  - All-Active - In all-active mode all the PEs attached to the particular Ethernet-Segment is allowed to forward traffic to and from that Ethernet Segment.

EVPN-VPWS Single Homed

The EVPN-VPWS single homed solution requires per EVI Ethernet Auto Discovery route. EVPN defines a new BGP Network Layer Reachability Information (NLRI) used to carry all EVPN routes. BGP Capabilities Advertisement used to ensure that two speakers support EVPN NLRI (AFI 25, SAFI 70) as per RFC 4760.

The architecture for EVPN VPWS is that the PEs run Multi-Protocol BGP in control-plane. The following image describes the EVPN-VPWS configuration:



- The VPWS service on PE1 requires the following three elements to be specified at configuration time:
  - The VPN ID (EVI)
  - The local AC identifier (AC1) that identifies the local end of the emulated service.
  - The remote AC identifier (AC2) that identifies the remote end of the emulated service.

  PE1 allocates a MPLS label per local AC for reachability.
- The VPWS service on PE2 is set in the same manner as PE1. The three same elements are required and the service configuration must be symmetric.

  PE2 allocates a MPLS label per local AC for reachability.
- PE1 advertise a single EVPN per EVI Ethernet AD route for each local endpoint (AC) to remote PEs with the associated MPLS label.

  PE2 performs the same task.
- On reception of EVPN per EVI EAD route from PE2, PE1 adds the entry to its local L2 RIB. PE1 knows the path list to reach AC2, for example, next hop is PE2 IP address and MPLS label for AC2.

  PE2 performs the same task.

Configure EVPN-VPWS Single Homed

This section describes how you can configure single-homed EVPN-VPWS feature.

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.10.10.1
Router(config-bgp-nbr))# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit
Router(config-bgp-nbr-af)# exit
Router(config-bgp-nbr)# exit
Router(config-bgp)# exit
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/1/0/2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 100 target 12 source 10
Router(config-l2vpn-xc-p2p)# commit
```

Running Configuration

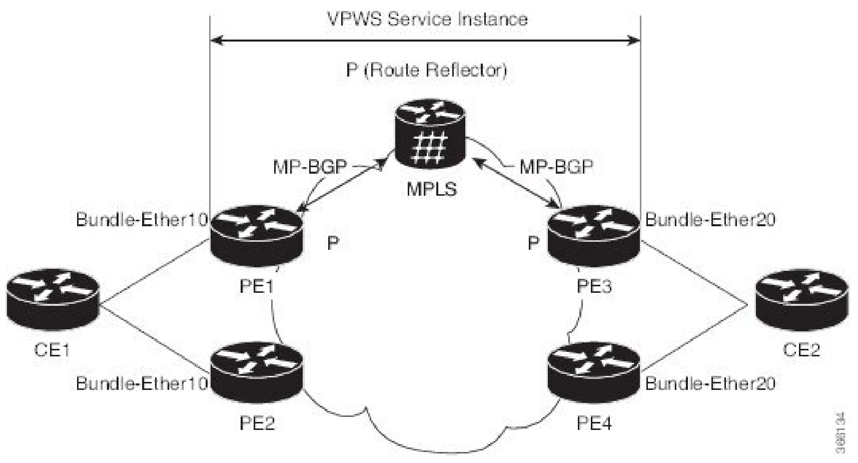
```
configure
router bgp 100
 address-family l2vpn evpn
  neighbor 10.10.10.1
   address-family l2vpn evpn
!

configure
l2vpn
 xconnect group evpn-vpws
  p2p evpn1
   interface TenGigE0/1/0/2
   neighbor evpn evi 100 target 12 source 10
!
```

EVPN-VPWS Multi-Homed

The EVPN VPWS feature supports all-active multihoming capability that enables you to connect a customer edge device to two or more provider edge (PE) devices to provide load balancing and redundant connectivity. The load balancing is done using equal-cost multipath (ECMP).

When a CE device is multi-homed to two or more PEs and when all PEs can forward traffic to and from the multi-homed device for the VLAN, then such multihoming is referred to as all-active multihoming.

Figure 102: EVPN VPWS Multi-Homed



Consider the topology in which CE1 is multi-homed to PE1 and PE2; CE2 is multi-homed to PE3 and PE4. PE1 and PE2 will advertise an EAD per EVI route per AC to remote PEs which is PE3 and PE4, with the associated MPLS label. The ES-EAD route is advertised per ES (main interface), and it will not have a label. Similarly, PE3 and PE4 advertise an EAD per EVI route per AC to remote PEs, which is PE1 and PE2, with the associated MPLS label.

Consider a traffic flow from CE1 to CE2. Traffic is sent to either PE1 or PE2. The selection of path is dependent on the CE implementation for forwarding over a LAG. Traffic is encapsulated at each PE and forwarded to the remote PEs (PE 3 and PE4) through MPLS core. Selection of the destination PE is established by flow-based load balancing. PE3 and PE4 send the traffic to CE2. The selection of path from PE3 or PE4 to CE2 is established by flow-based load balancing.

If there is a failure and when the link from CE1 to PE1 goes down, the PE1 withdraws the ES-EAD route; sends a signal to the remote PEs to switch all the VPWS service instances associated with this multi-homed ES to backup PE, which is PE2.

Configure EVPN-VPWS Single-Active Multi-Homed

This section describes how to configure single-active multi-homed EVPN-VPWS feature. You can enable the single-active mode by using the **load-balancing-mode single-active** command.

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)# exit

Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# commit


/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc-p2p)# exit

Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# commit


/* Configure PE3 */
Router# configure
```

```
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit

Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# commit


/* Configure PE4 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# load-balancing-mode single-active
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
/* On PE1 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00
   load-balancing-mode single-active
!


/* On PE2 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
```

```
   identifier type 0 00.01.00.ac.ce.55.00.0a.00
   load-balancing-mode single-active
!

/* On PE3 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00
      load-balancing-mode single-active
!

/* On PE4 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00
      load-balancing-mode single-active
!
```

Configure EVPN-VPWS All-Active Multi-Homed

This section describes how you can configure multi-homed EVPN-VPWS feature.

```
/* Configure PE1 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# commit

/* Configure PE2 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether10.2
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 5 source 6
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether10
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.0a.00
Router(config-evpn-ac-es)# commit

/* Configure PE3 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit

/* Configure PE4 */
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)#  xconnect group evpn_vpws
Router(config-l2vpn-xc)#  p2p e1_5-6
Router(config-l2vpn-xc-p2p)#  interface Bundle-Ether20.1
Router(config-l2vpn-xc-p2p)#  neighbor evpn evi 1 target 6 source 5
Router(config-l2vpn-xc-p2p)# exit
Router(config-l2vpn-xc)# exit
Router(config-l2vpn)# exit
Router(config)# evpn
Router(config-evpn)# interface Bundle-Ether20
Router(config-evpn-ac)#  ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.ce.55.00.14.00
Router(config-evpn-ac-es)# commit
```

Running Configuration

```
/* On PE1 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00

!


/* On PE2 */
!
configure
```

```
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether10.2
  neighbor evpn evi 1 target 5 source 6
!
evpn
interface Bundle-Ether10
  ethernet-segment
   identifier type 0 00.01.00.ac.ce.55.00.0a.00

!

/* On PE3 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00

!

/* On PE4 */
!
configure
l2vpn xconnect group evpn_vpws
 p2p e1_5-6
  interface Bundle-Ether20.1
  neighbor evpn evi 1 target 6 source 5
!
evpn
interface Bundle-Ether20
  ethernet-segment
      identifier type 0 00.01.00.ac.ce.55.00.14.00

!
```

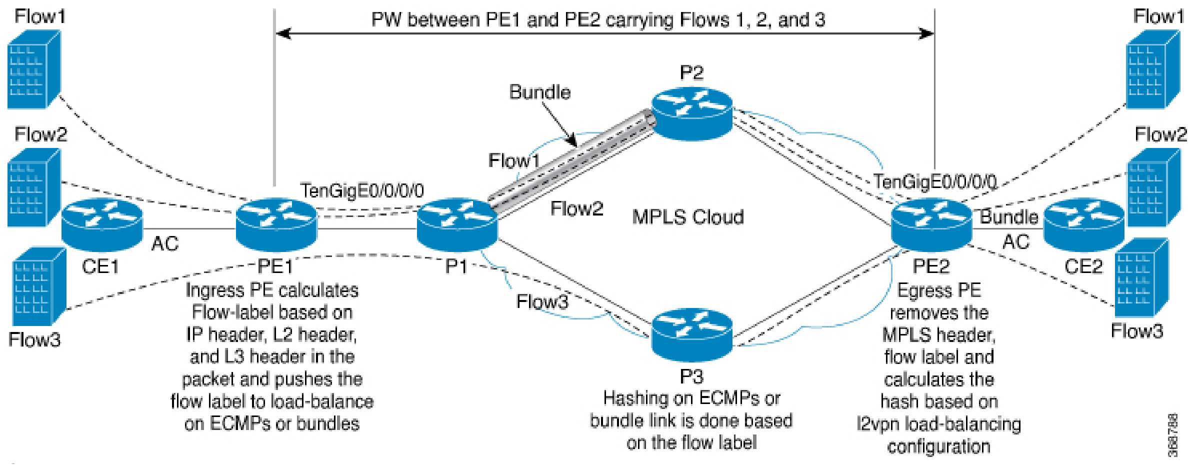
Flow Label Support for EVPN VPWS

The Flow Label support for EVPN VPWS feature enables provider (P) routers to use the flow-based load balancing to forward traffic between the provider edge (PE) devices. This feature uses Flow-Aware Transport (FAT) of pseudowires (PW) over an MPLS packet switched network for load-balancing traffic across BGP-based signaled pseudowires for Ethernet VPN (EVPN) Virtual Private Wire Service (VPWS).

FAT PWs provide the capability to identify individual flows within a PW and provide routers the ability to use these flows to load-balance the traffic. FAT PWs are used to load balance the traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on indivisible packet flows entering an imposition PE. This flow label is inserted as the lower most label in the packet. P routers use the flow label for load balancing to provide better traffic distribution across ECMP paths or link-bundled paths in the core. A flow is identified either by the source and destination IP address and layer 4 source and destination ports of the traffic, or the source and destination MAC address of the traffic.

The following figure shows a FAT PW with two flows distributing over ECMPs and bundle links.

Figure 103: FAT PW with Two Flows Distributing over ECMPs and Bundle


Flow1
PW between PE1 and PE2 carrying Flows 1, 2, and 3
Flow1
P2
Bundle
Flow2
Flow2
Flow1
TenGigE0/0/0/0
TenGigE0/0/0/0
Flow2
MPLS Cloud
Bundle
AC
AC
CE1
PE1
P1
PE2
CE2
Ingress PE calculates Flow-label based on IP header, L2 header, and L3 header in the packet and pushes the flow label to load-balance on ECMPs or bundles
Flow3
Egress PE removes the MPLS header, flow label and calculates the hash based on l2vpn load-balancing configuration
Flow3
Flow3
P3
Hashing on ECMPs or bundle link is done based on the flow label
368788

An extra label is added to the stack, called the flow label, which is generated for each unique incoming flow on the PE. A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set. The flow label is inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The FAT PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

Core routers perform load balancing using the flow-label in the FAT PW with other information like MAC address and IP address. The flow-label adds greater entropy to improve traffic load balancing. Therefore, it’s possible to distribute flows over ECMPs and link bundles.

In this topology, the imposition router, PE1, adds a flow label in the traffic. The disposition router, PE2, allows mixed types of traffic of which some have flow label, others do not. The P router uses flow label to load balance the traffic between the PEs. PE2 ignores the flow label in traffic, and uses one EVPN label for all unicast traffic.

Restrictions

To configure flow label for EVPN VPWS, the following restrictions are applicable:

- This feature is not supported for EVPN Point-to-Multipoint (P2MP) of VPLS and Ethernet LAN (E-LAN) service.
- This feature is supported only for EVPN VPWS single homing. AC bundle interfaces must be configured with ESI-0 only.
- This feature is not supported for EVPN flexible cross-connect service.

Configure Flow Label for EVPN VPWS

Configuration Example

Perform this task to configure flow label for EVPN VPWS on both PE1 and PE2.

```
Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn-vpws
Router(config-l2vpn-xc)# p2p evpn1
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/0/0
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1 target 2 source 1
Router(config-l2vpn-xc-p2p)# exit
!
Router# configure
Router(config)# evpn
Router(config-evpn)# evi 1
Router(config-evpn-instance)# control-word-disable
Router(config-evpn-instance)# load-balancing
Router(config-evpn-instance-lb)# flow-label static
Router(config-evpn-instance-lb)# commit
```

Running Configuration

This section shows the running configuration of flow label for EVPN VPWS.

```
l2vpn
 xconnect group evpn-vpws
  p2p evpn1
   interface TenGigE0/0/0/0
   neighbor evpn evi 1 target 2 source 1
   !
  !
evpn
 evi 1
  control-word-disable
  load-balancing
   flow-label static
  !
!
```

Verification

Verify EVPN VPWS flow label configuration.

```
Router# show l2vpn xconnect detail
Group evpn-vpws, XC evpn1, state is up; Interworking none
  AC: TenGigE0/0/0/0, state is up
    Type Ethernet
    MTU 1500; XC ID 0x1; interworking none
    Statistics:
      packets: received 21757444, sent 0
      bytes: received 18226521128, sent 0
  EVPN: neighbor 100.100.100.2, PW ID: evi 1, ac-id 2, state is up ( established )
    XC ID 0xc0000001
    Encapsulation MPLS
   Encap type Ethernet, control word disabled
    Sequencing not set
    LSP : Up
    Flow Label flags configured (Tx=1,Rx=1) statically

      EVPN         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        64002                          64002
      MTU          1500                           1500
      Control word disabled                       disabled
      AC ID        1                              2
      EVPN type    Ethernet                       Ethernet
      ------------ ------------------------------ -----------------------------
    Create time: 30/10/2018 03:04:16 (00:00:40 ago)
    Last time status changed: 30/10/2018 03:04:16 (00:00:40 ago)
    Statistics:
      packets: received 0, sent 21757444
      bytes: received 0, sent 18226521128
```

Related Topics

- Flow Label Support for EVPN VPWS, on page 764

Associated Commands

- show evpn evi

CHAPTER 14

L2VPN Services over Segment Routing for Traffic Engineering Policy

This chapter describes how to configure segment routing for traffic-engineering policies on the Cisco ASR 9000 Series Aggregation Services Routers supporting Cisco IOS XR software.

Overview of Segment Routing

Segment Routing (SR) is a flexible and scalable way of performing source routing. The source device selects a path and encodes it in the packet header as an ordered list of segments. Segments are identifiers for any type of instruction.

Segment routing for traffic engineering (SR-TE) takes place through a tunnel between a source and destination pair. SR-TE uses the concept of source routing, where the source calculates the path and encodes it in the packet header as a segment. Segments are identifier for any type of instruction. In SR-TE preferred path, each segment is an end-to-end path from the source to the destination, and instructs the routers in the provider core network to follow the specified path instead of the shortest path calculated by the IGP. The destination is unaware of the presence of the tunnel.

Each segment is identified by the segment ID (SID) consisting of a flat unsigned 32-bit integer. Segment instruction can be:

- Go to node N using the shortest path
- Go to node N over the shortest path to node M and then follow links Layer 1, Layer 2, and Layer 3
- Apply service S

The user can achieve better resilience and convergence for the network traffic, by transporting MPLS L2VPN services using segment routing, instead of MPLS LDP. Segment routing can be directly applied to the MPLS architecture without changing the forwarding plane. In a segment-routing network that uses the MPLS data plane, LDP or other signaling protocol is not required; instead label distribution is performed by IGP. Removing protocols from the network simplifies its operation and makes it more robust and stable by eliminating the need for protocol interaction. Segment routing utilizes the network bandwidth more effectively than traditional MPLS networks and offers lower latency.

With segment routing, the network no longer needs to maintain a per-application and per-flow state. Instead, it obeys the forwarding instructions provided in the packet.

Segment Routing relies on a small number of extensions to Cisco Intermediate System-to-Intermediate System (IS-IS) and Open Shortest Path First (OSPF) protocols. It can operate with an MPLS (Multiprotocol Label Switching) or an IPv6 data plane, and it integrates with the rich multi service capabilities of MPLS, including Layer 3 VPN (L3VPN), Virtual Private Wire Service (VPWS), Virtual Private LAN Service (VPLS), and Ethernet VPN (EVPN).

Segment routing can be directly applied to the Multiprotocol Label Switching (MPLS) architecture with no change in the forwarding plane. Segment routing utilizes the network bandwidth more effectively than traditional MPLS networks and offers lower latency. A segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels. The segment to process is on the top of the stack. The related label is popped from the stack, after the completion of a segment.

Segment Routing provides automatic traffic protection without any topological restrictions. The network protects traffic against link and node failures without requiring additional signaling in the network. Existing IP fast re-route (FRR) technology, in combination with the explicit routing capabilities in Segment Routing guarantees full protection coverage with optimum backup paths. Traffic protection does not impose any additional signaling requirements.

How Segment Routing Works

A router in a Segment Routing network is capable of selecting any path to forward traffic, whether it is explicit or Interior Gateway Protocol (IGP) shortest path. Segments represent subpaths that a router can combine to form a complete route to a network destination. Each segment has an identifier (Segment Identifier) that is distributed throughout the network using new IGP extensions. The extensions are equally applicable to IPv4 and IPv6 control planes. Unlike the case for traditional MPLS networks, routers in a Segment Router network do not require Label Distribution Protocol (LDP) and Resource Reservation Protocol - Traffic Engineering (RSVP-TE) to allocate or signal their segment identifiers and program their forwarding information.

There are two ways to configure segment routing:

- SR-TE policy under "segment-routing traffic-eng" sub-mode
- TE tunnel with SR option under "mpls traffic-eng" sub-mode

Note However, you can configure the above mentioned L2VPN and EVPN services using only "segment-routing traffic-eng" sub-mode.

Each router (node) and each link (adjacency) has an associated segment identifier (SID). Node segment identifiers are globally unique and represent the shortest path to a router as determined by the IGP. The network administrator allocates a node ID to each router from a reserved block. On the other hand, an adjacency segment ID is locally significant and represents a specific adjacency, such as egress interface, to a neighboring router. Routers automatically generate adjacency identifiers outside of the reserved block of node IDs. In an MPLS network, a segment identifier is encoded as an MPLS label stack entry. Segment IDs direct the data along a specified path. There are two kinds of segment IDS:

- Prefix SID: A segment ID that contains an IP address prefix calculated by an IGP in the service provider core network. Prefix SIDs are globally unique. A prefix segment represents the shortest path (as computed by IGP) to reach a specific prefix; a node segment is a special prefix segment that is bound to the loopback address of a node. It is advertised as an index into the node specific SR Global Block or SRGB.
- Adjacency SID: A segment ID that contains an advertising router's adjacency to a neighbor. An adjacency SID is a link between two routers. Since the adjacency SID is relative to a specific router, it is locally unique.

  A node segment can be a multi-hop path while an adjacency segment is a one-hop path.

Segment Routing Global Block

Segment Routing Global Block (SRGB) is the range of labels reserved for segment routing. SRGB is local property of an segment routing node. In MPLS, architecture, SRGB is the set of local labels reserved for global segments. In segment routing, each node can be configured with a different SRGB value and hence the absolute SID value associated to an IGP Prefix Segment can change from node to node.

The SRGB default value is 16000 to 23999. The SRGB can be configured as follows:

```
Router(config)# router isis 1
Router(config-isis)#segment-routing global-block 45000 55000
```

Preferred Path over SR-TE

Preferred tunnel path functionality allows you map pseudowires to specific traffic-engineering tunnel paths. Attachment circuits are cross-connected to specific SR traffic engineering tunnel interfaces instead of remote PE router IP addresses reachable using IGP or LDP. Using preferred tunnel path, the traffic engineering tunnel transports L2 or L3 traffic between the source and destination PE routers. A path is selected for an SR Policy when the path is valid and its preference is the best (highest value) among all the candidate paths of the SR Policy.

EVPN VPWS Preferred Path over SR-TE Policy

EVPN VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for EVPN VPWS pseudowire (PW) using SR-TE policy. SR policy allows you to choose the path on a per EVPN instance (EVI) basis. This feature is supported on bundle attachment circuit (AC), physical AC, and PWHE (PW-Ether interfaces) AC.

Restrictions

- If EVPN VPWS with On Demand Next Hop (ODN) is configured, and EVPN VPWS with preferred path is also configured for the same PW, then the preferred-path will take precedence.
- EVPN VPWS SR policy is not supported on EVPN VPWS dual homing.
- L2VPN services over PWHE with EVPN VPWS preferred-path over SR-TE policy is not supported.
- PWHE over SR-TE with Equal-cost multipath (ECMP) paths is not supported.

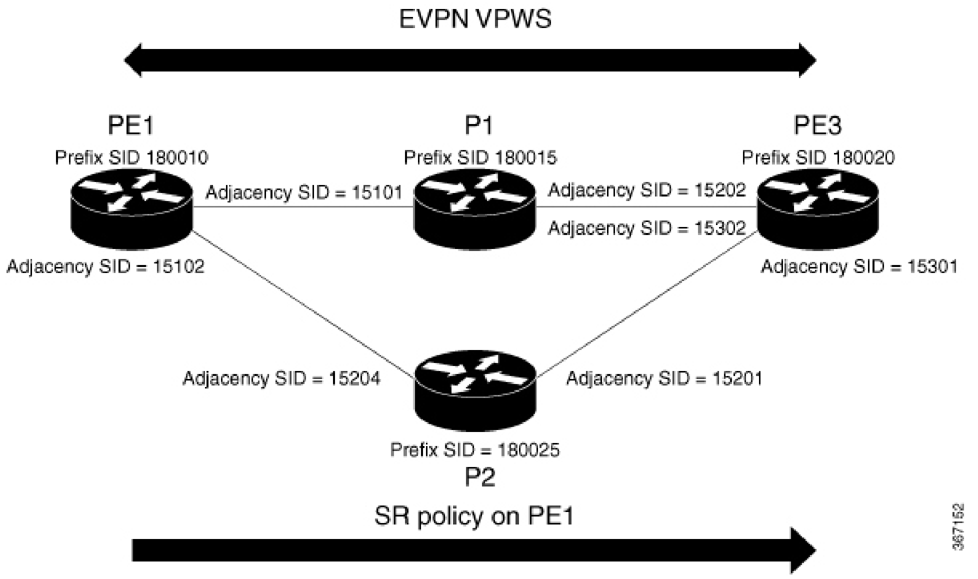
- EVPN validates if the route is for a single home next hop, otherwise it issues an error message about a dangling SR TE policy, and continue to set up EVPN-VPWS without it. EVPN relies on ESI value being zero to determine if this is a single home or not. If the AC is a Bundle-Ether interface running LACP then you need to manually configure the ESI value to zero to overwrite the auto-sense ESI as EVPN VPWS multihoming is not supported.

  To disable EVPN dual homing, configure bundle-Ether AC with ESI value set to zero.

```
evpn
interface Bundle-Ether12
  ethernet-segment
    identifier type 0 00.00.00.00.00.00.00.00.00
/* Or globally */
Evpn
 ethernet-segment type 1 auto-generation-disable
```

Topology

Figure 104: EVPN VPWS Preferred Path over SR-TE Policy



Consider a topology where PE1 and PE3 are the two EVPN VPWS PW end-points. Traffic is sent from PE1 to PE3 through SR in the core. Traffic from PE1 can be sent to PE3 either through P1 or P2 node. In this example, the EVPN VPWS preferred path over SR policy is configured to show the traffic flow from PE1 to PE3 using prefix-SID. Using adjacency-SID, you can steer traffic flow from PE1 to PE3 and specify whether it should pass through P1 or P2 node.

Configure EVPN VPWS Preferred Path over SR-TE Policy

You must complete these tasks to ensure the successful configuration of EVPN VPWS Preferred Path over SR-TE Policy feature:

- Configure Prefix-SID on IGP — The following examples show how to configure prefix-SID in IS-IS.
- Configure Adjacency-SID on IGP — The following examples show how to configure Adjacency-SID in IS-IS.

- Configure segment-list
- Configure SR-TE policy
- Configure EVPN VPWS over SR-TE policy
- Configure VPWS pseudowire headend over SR-TE policy

Configure Prefix-SID in IS-IS

Configure Prefix-SID on PE1, P1, P2, and PE3.

```
/* Configure Prefix-SID on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0031.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id 1.1.1.1
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180010
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on P1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Router(config)# router isis core
Router(config-isis)# is-type level-2-only
Router(config-isis)# net 49.0002.0330.2000.0021.00
Router(config-isis)# nsr
Router(config-isis)# nsf ietf
Router(config-isis)# log adjacency changes
Router(config-isis)# address-family ipv4 unicast
Router(config-isis-af)# metric-style wide level 2
Router(config-isis-af)# mpls traffic-eng level-2-only
Router(config-isis-af)# mpls traffic-eng router-id loopback0
Router(config-isis-af)# segment-routing mpls sr-prefer
Router(config-isis-af)# segment-routing prefix-sid-map advertise-local
Router(config-isis-af)# exit
```

```
!
Router(config-isis)# interface loopback 0
Router(config-isis-if)# address-family ipv4 unicast
Router(config-isis-af)# prefix-sid index 180015
Router(config-isis-af)# commit
Router(config-isis-af)# exit

/* Configure Prefix-SID on P2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block  180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0022.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180025
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.3030.0030.0035.00
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 180020
Route(config-isis-af)# commit
Route(config-isis-af)# exit
```

Configure Adjacency-SID in IS-IS

Configure Adjacency-SID on PE1, P1, P2, and PE3.

```
/* Configure Adjacency-SID on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15101
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/1/6
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15102
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on P1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15200
Route(config-isis-if-af)# commit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15202
Route(config-isis-if-af)# commit
!

/* Configure Adjacency-SID on P2 */

Router# configure
```

```
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15201
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/5
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# metric 20
Route(config-isis-if-af)# adjacency-sid absolute 15204
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block 15000 15999
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/1
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15301
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/2
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15302
Route(config-isis-if-af)# commit
```

Configure Segment-list

```
/* Configure Segment-list on PE1 using prefix-SID */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# traffic-eng
Router(config-sr-te)# logging
Router(config-sr-te-log)# policy status
Router(config-sr-te-log)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name pref_sid_to_PE3
Router(config-sr-te-sl)# index 1 mpls label 180020 <------------------------using prefix-SID
Router(config-sr-te-sl)# exit

/* Configure Segment-list on PE1 using adjacency-SID */

Router# configure
Router(config)# segment-routing
Router(config-sr)# local-block  15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# logging
Router(config-sr-te-log)# policy status
Router(config-sr-te-log)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list name pref_adj_sid_to_PE3
Router(config-sr-te-sl)# index 1 mpls label 15101 <------------------------using adjacency-SID
Router(config-sr-te-sl)# index 2 mpls label 15202 <------------------------using adjacency-SID
Router(config-sr-te-sl)# exit
```

Configure SR-TE Policy

```
/* Configure SR-TE Policy */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy pref_sid_to_PE3
Router(config-sr-te-policy)# color 9001 end-point ipv4 20.20.20.20
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 10
Router(config-sr-te-pp-info)# explicit segment-list pref_sid_to_PE3
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy pref_adj_sid_to_PE3
Router(config-sr-te-policy)# color 9001 end-point ipv4 20.20.20.20
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list pref_adj_sid_to_PE3
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit

/* You can configure multiple preferences for an SR policy. Among the configured preferences,
 the largest number takes the highest precedence */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 180000 200000
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1013
Router(config-sr-te-policy)# color 1013 end-point ipv4 2.2.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-P1_BE121
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-P1-t0016
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 700 <----------largest number takes the precendence
Router(config-sr-te-pp-info)# explicit segment-list PE1-P1
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit
```

Configure EVPN VPWS over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name pref_sid_to_PE3

SR-TE policy database
---------------------
Color: 9001, End-point: 20.20.20.20
 Name: srte_c_9001_ep_20.20.20.20

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class 1001
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_9001_ep_20.20.20.20
fallback disable
Router(config-l2vpn-pwc-mpls)# commit
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p evpn_vpws_1001
Router(config-l2vpn-xc-p2p)# interface tengi0/1/0/1.1001
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 1001 target 10001 source 20001
Router(config-l2vpn-xc-p2p-pw)# pw-class 1001
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# exit



/* If Fallback Enable is configured, which is the default option, and if the SR-policy is
down, then EVPN VPWS will still continue to be UP using the regular IGP path, and not using
 the SR-policy */
show l2vpn xconnect detail
  EVPN: neighbor 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is up ( established )
    Preferred path Inactive : SR TE srte_c_9001_ep_20.20.20.20, Statically configured,
fallback enabled
    Tunnel : Down
    LSP: Up
```

```
/* If Fallback Disable is configured, and if the SR-policy is down, or if it misconfigured
 in dual homed mode, then the L2VPN PW will be down */
show l2vpn xconnect detail
EVPN: neighbor 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is down ( local ready )
Preferred path Active : SR TE srte_c_9001_ep_20.20.20.20, Statically configured, fallback
disabled
Tunnel : Down
```

Configure VPWS Pseudowire Headend over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name 1013

SR-TE policy database
---------------------
Color: 1013, End-point: 2.2.2.2
 Name: srte_c_1013_ep_2.2.2.2


Router# configure
Router(config)# interface PW-Ether1
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# attach generic-interface-list 1
Router(config-if)# exit
!
Router(config)# generic-interface-list 1
Router(config-gen-if-list)# interface Bundle-Ether1
Router(config-gen-if-list)# interface TenGigE0/1/0/6/1
Router(config-gen-if-list)# interface TenGigE0/1/0/6/2
Router(config-gen-if-list)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pw1013
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_1013_ep_2.2.2.2 fallback
 disable
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# Xconnect group evpn_vpws_pwhe
Router(config-l2vpn-xc)# p2p 1001
Router(config-l2vpn-xc-p2p)# interface pw-ether 1
Router(config-l2vpn-xc-p2p-pw)# neighbor evpn evi 1000 target 1000 source 1000
Router(config-l2vpn-xc-p2p-pw)# pw-class pw1013
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* Configure Prefix-SID in ISIS */
PE1:

configure
```

```
 segment-routing
  global-block 180000 200000
!
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0031.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id 1.1.1.1
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180010
```

P1:

```
configure
 segment-routing
  global-block 180000 200000

router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0021.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180015
```

P2:

```
configure
 segment-routing
  global-block 180000 200000

router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0022.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

 interface Loopback0
  address-family ipv4 unicast
```

```
   prefix-sid index 180025

PE3:

configure
 segment-routing
  global-block 180000 200000

router isis core
 is-type level-2-only
 net 49.0002.0330.2000.3030.0030.0035.00
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 180020

/* Configure Adjacency-SID in ISIS */

PE1:

configure
 segment-routing
  local-block 15000 15999
!

router isis core
 !
interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15101

interface TenGigE0/0/1/6
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15102

P1:

configure
 segment-routing
  local-block 15000 15999

router isis core
 !
interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15200

interface TenGigE0/0/0/0/7
```

```
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15202
```

P2:

```
configure
 segment-routing
  local-block 15000 15999

router isis core
 !
interface TenGigE0/0/0/5
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15204

interface TenGigE0/0/0/0/7
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   metric 20
   adjacency-sid absolute 15201
```

PE3:

```
configure
 segment-routing
  local-block 15000 15999

router isis core
  !
interface TenGigE0/0/0/1
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15301
  !
 !
 interface TenGigE0/0/0/2
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15302

/* Configure Segment-list */
```

PE1:

```
configure
 segment-routing
  global-block 180000 200000
  traffic-eng
   logging
    policy status
```

```
segment-routing
 traffic-eng
  segment-list name pref_sid_to_PE3
   index 1 mpls label 180020
  !
 !
configure
 segment-routing
  local-block 15000 15999
  traffic-eng
   logging
    policy status

segment-routing
 traffic-eng
  segment-list name pref_adj_sid_to_PE3
   index 1 mpls label 15101
   index 2 mpls label 15202
  !
 !

/* Configure SR-TE policy */

segment-routing
 traffic-eng
  policy pref_sid_to_PE3
   color 9001 end-point ipv4 20.20.20.20
   candidate-paths
    preference 10
     explicit segment-list pref_sid_to_PE3
     !
    !
segment-routing
 traffic-eng
  policy pref_adj_sid_to_PE3
   color 9001 end-point ipv4 20.20.20.20
   candidate-paths
    preference 200
     explicit segment-list pref_adj_sid_to_PE3
     !
    !

/* You can configure multiple preferences for an SR policy. Among the configured preferences,
 the largest number takes the highest precedence */

segment-routing
 traffic-eng
  policy 1013
   color 1013 end-point ipv4 2.2.2.2
   candidate-paths
    preference 100
     explicit segment-list PE1-P1_BE121
     !
    preference 200
     explicit segment-list PE1-PE3-P1-t0016
     !
    preference 700
     explicit segment-list PE1-P1
     !

/* Configure EVPN VPWS over SR-TE policy */
PE1:
configure
 l2vpn
```

```
  pw-class 1001
   encapsulation mpls
    preferred-path sr-te policy srte_c_9001_ep_20.20.20.20 fallback disable
xconnect group evpn_vpws
  p2p evpn_vpws_1001
   interface tengi0/1/0/1.1001
   neighbor evpn evi 1001 target 10001 source 20001
    pw-class 1001
   !

/* Configure VPWS Pseudowire Headend SR-TE Policy */

interface PW-Ether1
ipv4 address 10.1.1.1 255.255.255.0
attach generic-interface-list 1

generic-interface-list 1
interface Bundle-Ether1
interface TenGigE0/1/0/6/1
interface TenGigE0/1/0/6/2



l2vpn
 pw-class pw1013
  encapsulation mpls
   transport-mode ethernet
   preferred-path sr-te policy srte_c_1013_ep_2.2.2.2 fallback disable
Xconnect group evpn_vpws_pwhe
  p2p 1001
   interface pw-ether 1
   neighbor evpn evi 1000 target 1000 source 1000
     pw-class pw1013
```

Verify EVPN VPWS Preferred Path over SR-TE Policy Configuration

```
PE1#show segment-routing traffic-eng forwarding policy name pref_sid_to_PE3 detail
Policy        Segment         Outgoing    Outgoing            Next Hop        Bytes
Name          List            Label       Interface                           Switched
------------- --------------- ----------- ------------------- --------------- ------------
pref_sid_to_PE3

                              15102       TenGigE0/0/1/6       20.20.20.20       81950960
                 Label Stack (Top -> Bottom): { 15101, 15102 }
                 Path-id: 1, Weight: 0
                 Packets Switched: 787990
   Local label: 34555
   Packets/Bytes Switched: 1016545/105720680
   (!): FRR pure backup


PE1#show segment-routing traffic-eng policy candidate-path name pref_sid_to_PE3

SR-TE policy database
---------------------

Color: 9001, End-point: 20.20.20.20
  Name: srte_c_9001_ep_20.20.20.20


PE1#show mpls forwarding tunnels sr-policy name pref_sid_to_PE3
Tunnel        Outgoing    Outgoing     Next Hop        Bytes
```

```
Name          Label       Interface                    Switched
------------- ----------- ------------ --------------- ------------
pref_sid_to_PE3 (SR) 15102 TenGigE0/0/1/6 20.20.20.20       836516512


PE1#show l2vpn xconnect group evpn_vpws xc-name evpn_vpws_1001 detail
Group evpn_vpws, XC evpn_vpws_1001, state is up; Interworking none
  AC: Bundle-Ether12.1001, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 1000
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1500; XC ID 0xc0000018; interworking none
    Statistics:
      packets: received 642304, sent 642244
      bytes: received 61661184, sent 61655424
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is up ( established )
    XC ID 0xa0000007
    Encapsulation MPLS
    Source address 10.10.10.10
    Encap type Ethernet, control word enabled
    Sequencing not set
    Preferred path Active : SR TE pref_sid_to_PE3, Statically configured, fallback disabled
    Tunnel : Up
    Load Balance Hashing: src-dst-mac

PE# show l2vpn xconnect group evpn_vpws_pwhe xc-name  3001 detail
Group evpn_vpws_pwhe, XC 3001, state is up; Interworking none
  AC: PW-Ether3001, state is up
    Type PW-Ether
    Interface-list: GIL
      Internal label: 24022
  ….
  EVPN: neighbor 20.20.20.20, PW ID: evi 3001, ac-id 3001, state is up ( established )
   ….
    Preferred path Active : SR TE evpn_to_PE5_3001, Statically configured, fallback disabled
    Tunnel : Up

PE1# show mpls forwarding labels 24022 detail
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24022  25022       PW-HE(PE3001)      evpn_to_PE5_ point2point     0
     Updated: Dec 18 12:52:50.764
     PW HE VC type   : 5
     PW HE CW        : enabled
     PW HE interface : PE3001
     Outgoing Interface: evpn_to_PE5_3001 (ifhandle 0x0000fc20)

PE1# show l2vpn forwarding main-port pwhe location 0/2/CPU0
PWHE-MP Interface: PW-Ether3001
  ifhandle:             0x000101e0
  Generic Iflist ID:    1
  Internal_label:       24022
  Remote Label:         25022
  Control Word Enabled: Yes
  Pseudowire Type:      5
  Next Hop Tunnel IFH: 0xfc20
```

Associated Commands

- adjacency-sid

- index
- prefix-sid
- router isis
- segment-routing

The applicable segment routing commands are described in the *Segment Routing Command Reference for Cisco ASR 9000 Series Routers*

Related Topics

- Overview of Segment Routing , on page 769
- How Segment Routing Works , on page 770
- Segment Routing Global Block , on page 771

L2VPN VPLS or VPWS Preferred Path over SR-TE Policy

L2VPN VPLS or VPWS Preferred Path over SR-TE Policy feature allows you to set the preferred path between the two end-points for L2VPN Virtual Private LAN Service (VPLS) or Virtual Private Wire Service (VPWS) using SR-TE policy.

This feature is supported on bundle attachment circuit (AC), physical AC, and PWHE AC.

Restrictions

- When an SR policy is configured as the preferred path for a VPLS circuit, the traffic traverses through the SR policy path.

  The PW counters are updated with statistics about packets transmitted and received.

  When the SR policy configuration is deleted, the traffic session is still functional because the traffic transmission switches back to the normal LSP path between the PEs. There is no drop in the end-to-end traffic transmitted.

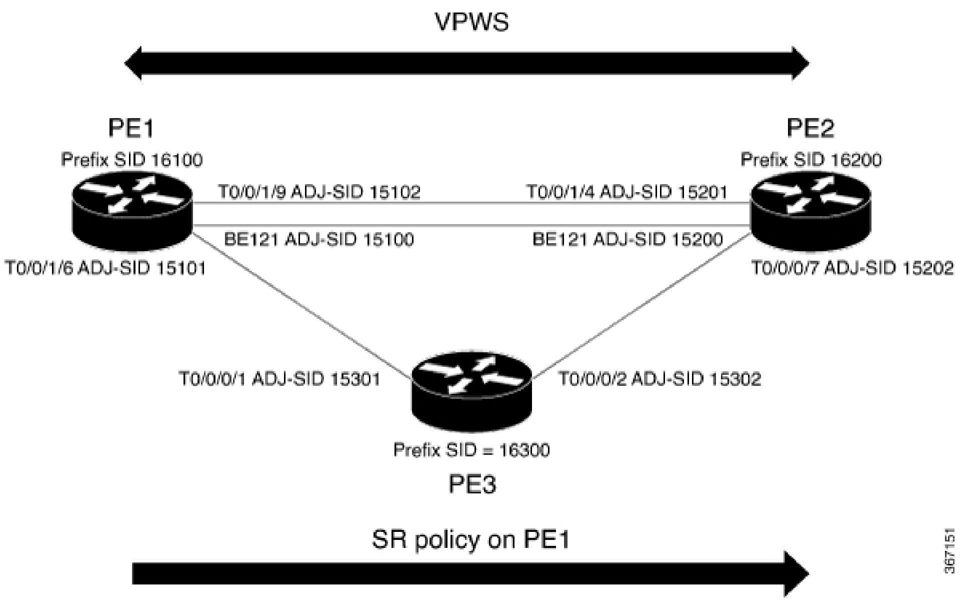
  However, the packet statistics counters are reset and start from zero.

  This is because, when the SR policy is deleted, the PW too gets deleted and the statistics information associated with the old PW is cleared. The counter restarts from zero when the new PW is created after the switch takes place.

Configure L2VPN VPLS or VPWS Preferred Path over SR-TE Policy

Perform the following steps to configure L2VPN VPLS or VPWS Preferred Path over SR-TE Policy feature. The following figure is used as a reference to explain the configuration steps.

Figure 105: L2VPN VPWS and VPLS Preferred Path over SR-TE Policy



- Configure Prefix-SID on IGP — The following examples show how to configure prefix-SID in IS-IS.
- Configure Adjacency-SID on IGP — The following examples show how to configure Adjacency-SID in IS-IS.
- Configure segment-list
- Configure SR-TE policy
- Configure VPLS over SR-TE policy
- Configure VPWS over SR-TE policy
- Configure VPWS pseudowire headend over SR-TE policy

Configure Prefix-SID in IS-IS

Configure Prefix-SID on PE1, PE2, and PE3.

```
/* Configure Prefix-SID on PE1 */

Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0031.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id 1.1.1.1
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
```

```
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16100
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE2 */

Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.0021.00
Route(config-isis)# nsr
Route(config-isis)# nsf ietf
Route(config-isis)# log adjacency changes
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16200
Route(config-isis-af)# commit
Route(config-isis-af)# exit

/* Configure Prefix-SID on PE3 */

Router# configure
Route(config)# router isis core
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0002.0330.2000.3030.0030.0035.00
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide level 2
Route(config-isis-af)# mpls traffic-eng level-2-only
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls sr-prefer
Route(config-isis-af)# segment-routing prefix-sid-map advertise-local
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback 0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 16300
Route(config-isis-af)# commit
Route(config-isis-af)# exit
```

Configure Adjacency-SID in IS-IS

Configure Adjacency-SID on PE1, PE2, and PE3.

```
/* Configure Adjacency-SID on PE1 */

Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15100
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/1/6
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15101
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/1/9
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15102
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE2 */

Router# configure
Route(config)# router isis core
Route(config-isis)# interface Bundle-Ether121
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15200
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigET0/0/1/4
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15201
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/7
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15202
Route(config-isis-if-af)# commit

/* Configure Adjacency-SID on PE3 */

Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/1
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15301
Route(config-isis-if-af)# exit
!
Router# configure
Route(config)# router isis core
Route(config-isis)# interface TenGigE0/0/0/2
Route(config-isis-if)# circuit-type level-2-only
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-if-af)# adjacency-sid absolute 15302
Route(config-isis-if-af)# commit
```

Configure Segment-list

Configure segment-list on PE1, PE2, and PE3.

```
/* Configure segment-list on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE1-PE2
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE2-PE3
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# index 2 mpls label 16300
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE2_bad
Router(config-sr-te-sl)# index 1 mpls label 16900
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16200
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE2_BE121
Router(config-sr-te-sl)# index 1 mpls label 15100
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2_link
Router(config-sr-te-sl)# index 1 mpls label 15101
Router(config-sr-te-sl)# index 2 mpls label 15302
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE1-PE3-PE2-t0016
Router(config-sr-te-sl)# index 1 mpls label 15101
Router(config-sr-te-sl)# index 2 mpls label 16200
Router(config-sr-te-sl)# commit


/* Configure segment-list on PE2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE2-PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16300
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit

/* Configure segment-list on PE3 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# global-block 16000 23999
Router(config-sr)# local-block 15000 15999
Router(config-sr)# traffic-eng
Router(config-sr-te)# segment-list segment-list name PE3-PE1
Router(config-sr-te-sl)# index 1 mpls label 16100
Router(config-sr-te-sl)# exit
!
Router(config-sr-te)# segment-list segment-list name PE3-PE2-PE1
Router(config-sr-te-sl)# index 1 mpls label 16200
Router(config-sr-te-sl)# index 2 mpls label 16100
Router(config-sr-te-sl)# commit
```

Configure SR-TE Policy

```
/* Configure SR-TE policy */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 100
Router(config-sr-te-policy)# color 1 end-point ipv4 2.2.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 400
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 500 <-----------------largest number takes the
precedence
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2
Router(config-sr-te-pp-info)# commit
Router(config-sr-te-pp-info)# exit


Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1013
Router(config-sr-te-policy)# color 1013 end-point ipv4 2.2.2.2
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2_BE121
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 200
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2-t0016
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 500
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE2
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 600
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2
Router(config-sr-te-pp-info)# exit
!
Router(config-sr-te-policy)# preference 700
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3-PE2_link
Router(config-sr-te-pp-info)# commit
!
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# policy 1300
Router(config-sr-te-policy)# color 1300 end-point ipv4 3.3.3.3
Router(config-sr-te-policy)# candidate-paths
Router(config-sr-te-policy)# preference 100
Router(config-sr-te-pp-info)# explicit segment-list PE1-PE3
Router(config-sr-te-pp-info)# commit
!
```

Configure VPLS over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name 100

SR-TE policy database
---------------------
Color: 1, End-point: 2.2.2.2
 Name: srte_c_1_ep_2.2.2.2

Router# show segment-routing traffic-eng policy candidate-path name 1013

SR-TE policy database
---------------------
Color: 1013, End-point: 2.2.2.2
 Name: srte_c_1013_ep_2.2.2.2

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pw100
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_1_ep_2.2.2.2
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config-l2vpn)# pw-class pw1013
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_1013_ep_2.2.2.2 fallback
 disable
Router(config-l2vpn-pwc-mpls)# exit

/* The default is Fallback Enable. If the SR-policy is down, then L2VPN VPWS/VPLS will try
 to be UP using the regular IGP path, and not using the SR policy. If Fallback Disable is
configured, the L2VPN PW will be down when the SR-policy is down. Preferred-path is the
action of pinning down a PW to a SR TE policy */


Router(config)#l2vpn
Router(config-l2vpn)#bridge group bg1
Router(config-l2vpn-bg)#bridge-domain vpls501
Router(config-l2vpn-bg-bd)#interface Bundle-Ether41.501
Router(config-l2vpn-bg-bd-ac)#exit
!
Router(config-l2vpn-bg-bd)#interface TenGigE0/0/1/0.501
Router(config-l2vpn-bg-bd-ac)#split-horizon group
Router(config-l2vpn-bg-bd-ac)#exit
!
Router(config-l2vpn-bg-bd)#vfi vpls1
Router(config-l2vpn-bg-bd-vfi)#neighbor 2.2.2.2 pw-id 501
Router(config-l2vpn-bg-bd-vfi-pw)#pw-class pw100
Router(config-l2vpn-bg-bd-vfi-pw)#exit
!
Router(config-l2vpn-bg-bd-vfi)#neighbor 3.3.3.3 pw-id 501
Router(config-l2vpn-bg-bd-vfi-pw)#commit
```

Configure VPWS over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name 1300

SR-TE policy database
---------------------
Color: 1300, End-point: 3.3.3.3
 Name: srte_c_1300_ep_3.3.3.3

Router# configure
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pw1300
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# load-balancing
Router(config-l2vpn-pwc-mpls-load-bal)# flow-label both
Router(config-l2vpn-pwc-mpls-load-bal)# exit
!
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_1300_cp_3.3.3.3 fallback
 disable
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xcon1
Router(config-l2vpn-xc)# p2p vplw1002
Router(config-l2vpn-xc-p2p)# interface TenGigE0/0/1/1.1002
Router(config-l2vpn-xc-p2p)# neighbor 3.3.3.3 pw-id 1002
Router(config-l2vpn-xc-p2p-pw)# pw-class pw1300
Router(config-l2vpn-xc-p2p-pw)# commit
Router(config-l2vpn-xc-p2p-pw)# exit
```

Configure VPWS Pseudowire Headend over SR-TE Policy

Note Use the auto-generated SR-TE policy name to attach the policy to the L2VPN instance. The auto-generated policy name is based on the policy color and end-point. Use the **show segment-routing traffic-eng policy candidate-path name** *policy_name* command to display the auto-generated policy name.

```
Router# show segment-routing traffic-eng policy candidate-path name 1013

SR-TE policy database
---------------------
Color: 1013, End-point: 2.2.2.2
 Name: srte_c_1013_ep_2.2.2.2


Router# configure
Router(config)# interface PW-Ether1
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# attach generic-interface-list 1
Router(config-if)# exit
!
Router(config)# generic-interface-list 1
Router(config-gen-if-list)# interface Bundle-Ether1
Router(config-gen-if-list)# interface TenGigE0/1/0/6/1
Router(config-gen-if-list)# interface TenGigE0/1/0/6/2
Router(config-gen-if-list)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# pw-class pw1013
Router(config-l2vpn-pwc)# encapsulation mpls
Router(config-l2vpn-pwc-mpls)# transport-mode ethernet
Router(config-l2vpn-pwc-mpls)# preferred-path sr-te policy srte_c_1013_ep_2.2.2.2 fallback
 disable
Router(config-l2vpn-pwc-mpls)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group xg2
Router(config-l2vpn-xc)# p2p pwhe10001
Router(config-l2vpn-xc-p2p)# interface PW-Ether10001
Router(config-l2vpn-xc-p2p-pw)# neighbor ipv4 2.2.2.2 pw-id 10001
Router(config-l2vpn-xc-p2p-pw)# pw-class pw1013
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* Configure prefix-SID */
PE1:
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0031.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id 1.1.1.1
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 16100
```

PE2:

```
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.0021.00
 nsr
 nsf ietf
 log adjacency changes
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 16200
```

PE3:

```
router isis core
 is-type level-2-only
 net 49.0002.0330.2000.3030.0030.0035.00
 address-family ipv4 unicast
  metric-style wide level 2
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls sr-prefer
  segment-routing prefix-sid-map advertise-local

interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 16300

/* Configure Adjacency-SID */
```

PE1:

```
router isis core
!
 interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15100
  !
 interface TenGigE0/0/1/6
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15101
  !
 interface TenGigE0/0/1/9
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15102

PE2
router isis core
 !
interface Bundle-Ether121
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15200

interface TenGigE0/0/0/0/4
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15201

interface TenGigE0/0/0/0/7
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15202

PE3:
router isis core
!
interface TenGigE0/0/0/1
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15301
  !
 !
 interface TenGigE0/0/0/2
  circuit-type level-2-only
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   adjacency-sid absolute 15302

/* Configure segment-list */
PE1:
segment-routing
 global-block 16000 23999
 local-block 15000 15999
  traffic-eng
 segment-list name PE1-PE2
   index 1 mpls label 16200
  !
  segment-list name PE1-PE3
   index 1 mpls label 16300
  !
  segment-list name PE1-PE2-PE3
   index 1 mpls label 16200
   index 2 mpls label 16300
  !
  segment-list name PE1-PE2_bad
   index 1 mpls label 16900
  !
  segment-list name PE1-PE3-PE2
   index 1 mpls label 16300
   index 2 mpls label 16200
  !
  segment-list name PE1-PE2_BE121
   index 1 mpls label 15100
!
  segment-list name PE1-PE3-PE2_link
   index 1 mpls label 15101
   index 2 mpls label 15302
  !

  segment-list name PE1-PE3-PE2-t0016
   index 1 mpls label 15101
   index 2 mpls label 16200
```

```
PE2:
segment-routing
 global-block 16000 23999
 local-block 15000 15999
 traffic-eng
  segment-list name PE2-PE1
   index 1 mpls label 16100
  !
  segment-list name PE2-PE3-PE1
   index 1 mpls label 16300
   index 2 mpls label 16100

PE3:
segment-routing
 global-block 16000 23999
 local-block 15000 15999
 traffic-eng
  segment-list name PE3-PE1
   index 1 mpls label 16100
  !
  segment-list name PE3-PE2-PE1
   index 1 mpls label 16200
   index 2 mpls label 16100

/* Configure SR-TE policy */

segment-routing
 traffic-eng
  policy 100
   color 1 end-point ipv4 2.2.2.2
   candidate-paths
    preference 400
     explicit segment-list PE1-PE3-PE2
     !
    preference 500
     explicit segment-list PE1-PE2

 policy 1013
   color 1013 end-point ipv4 2.2.2.2
   candidate-paths
    preference 100
     explicit segment-list PE1-PE2_BE121
     !
    preference 200
     explicit segment-list PE1-PE3-PE2-t0016
     !
    preference 500
     explicit segment-list PE1-PE2
      !
    preference 600
     explicit segment-list PE1-PE3-PE2
     !
    preference 700
     explicit segment-list PE1-PE3-PE2_link
     !
 policy 1300
   color 1300 end-point ipv4 3.3.3.3
   candidate-paths
    preference 100
     explicit segment-list PE1-PE3
     !

/* Configure VPLS over SR-TE policy
l2vpn
 pw-class pw100
  encapsulation mpls
   preferred-path sr-te policy srte_c_1_ep_2.2.2.2
 pw-class pw1013
  encapsulation mpls
   preferred-path sr-te policy srte_c_1013_ep_2.2.2.2 fallback disable
l2vpn
 bridge group bg1
  bridge-domain vpls501
   interface Bundle-Ether41.501
   !
   interface TenGigE0/0/1/0.501
    split-horizon group
   !
   vfi vpls1
    neighbor 2.2.2.2 pw-id 501
     pw-class pw100
    !
    neighbor 3.3.3.3 pw-id 501

/*Configure VPWS over SR-TE policy
l2vpn
 pw-class pw1013
  encapsulation mpls
    load-balancing
    flow-label both
   preferred-path sr-te policy srte_c_1013_ep_2.2.2.2 fallback disable

Xconnect group xcon1
  p2p vplw1002
   interface TenGigE0/0/1/1.1002
   neighbor 3.3.3.3 pw-id 1002
            pw-class pw1013

/* Configure VPWS Pseudowire Headend SR-TE Policy */

interface PW-Ether1
ipv4 address 10.1.1.1 255.255.255.0
attach generic-interface-list 1

generic-interface-list 1
interface Bundle-Ether1
interface TenGigE0/1/0/6/1
interface TenGigE0/1/0/6/2


l2vpn
 pw-class pw1300
  encapsulation mpls
   transport-mode ethernet
   preferred-path sr-te policy srte_c_1300_cp_3.3.3.3 [fallback disable]

l2vpn
 xconnect group xg2
  p2p pwhe10001
   interface PW-Ether10001
   neighbor ipv4 2.2.2.2 pw-id 10001
    pw-class pw1300
```

Verify L2VPN VPLS or VPWS Preferred Path over SR-TE Policy Configuration

```
/* The prefix-sid and Adjacency-sid must be in the SR topology */

PE1#show segment-routing traffic-eng ipv4 topology | inc Prefix
Thu Feb  1 20:28:43.343 EST
  Prefix SID:
    Prefix 1.1.1.1, label 16100 (regular)
  Prefix SID:
    Prefix 3.3.3.3, label 16300 (regular)
  Prefix SID:
    Prefix 2.2.2.2, label 16200 (regular)


PE1#show segment-routing traffic-eng ipv4 topology | inc Adj SID
Thu Feb  1 20:30:25.760 EST
    Adj SID: 61025 (unprotected) 15102 (unprotected)
    Adj SID: 61023 (unprotected) 15101 (unprotected)
    Adj SID: 65051 (unprotected) 15100 (unprotected)
    Adj SID: 41516 (unprotected) 15301 (unprotected)
    Adj SID: 41519 (unprotected) 15302 (unprotected)
    Adj SID: 46660 (unprotected) 15201 (unprotected)
    Adj SID: 24003 (unprotected) 15202 (unprotected)
    Adj SID: 46675 (unprotected) 15200 (unprotected)

PE1# show segment-routing traffic-eng policy candidate-path name 100

SR-TE policy database
---------------------

Color: 100, End-point: 2.2.2.2
 Name: srte_c_1_ep_2.2.2.2



PE1#show segment-routing traffic-eng policy name 100
Thu Feb  1 23:16:58.368 EST

SR-TE policy database
---------------------

Name: 100 (Color: 1, End-point: 2.2.2.2)
  Status:
    Admin: up  Operational: up for 05:44:25 (since Feb  1 17:32:34.434)
  Candidate-paths:
    Preference 500:
      Explicit: segment-list PE1-PE2 (active)
        Weight: 0, Metric Type: IGP
          16200 [Prefix-SID, 2.2.2.2]
    Preference 400:
      Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
  Attributes:
    Binding SID: 27498
      Allocation mode: dynamic
      State: Programmed
      Policy selected: yes
    Forward Class: 0

PE1#show segment-routing traffic-eng policy name 1013
Thu Feb  1 21:20:57.439 EST

SR-TE policy database
---------------------
Name: 1013 (Color: 1013, End-point: 2.2.2.2)
  Status:
    Admin: up  Operational: up for 00:06:36 (since Feb  1 21:14:22.057)
  Candidate-paths:
    Preference 700:
      Explicit: segment-list PE1-PE3-PE2_link (active)
        Weight: 0, Metric Type: IGP
          15101 [Adjacency-SID, 13.1.1.1 - 13.1.1.2]
          15302
    Preference 600:
      Explicit: segment-list PE1-PE3-PE2 (inactive)
      Inactive Reason:
        Weight: 0, Metric Type: IGP
    Preference 500:
      Explicit: segment-list PE1-PE2 (inactive)
      Inactive Reason:
        Weight: 0, Metric Type: IGP
    Preference 200:
      Explicit: segment-list PE1-PE3-PE2-t0016 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
    Preference 100:
      Explicit: segment-list PE1-PE2_BE121 (inactive)
      Inactive Reason: unresolved first label
        Weight: 0, Metric Type: IGP
  Attributes:
    Binding SID: 27525
      Allocation mode: dynamic
      State: Programmed
      Policy selected: yes
    Forward Class: 0

PE1#show segment-routing traffic-eng forwarding policy name 100
Thu Feb  1 23:19:28.951 EST
Policy        Segment         Outgoing    Outgoing            Next Hop        Bytes
Name          List            Label       Interface                           Switched
------------- --------------- ----------- ------------------- --------------- ------------
100           PE1-PE2         Pop         Te0/0/1/9           12.1.9.2        0
                                         Pop         BE121              121.1.0.2       0


PE1#show segment-routing traffic-eng forwarding policy name 1013 detail
Thu Feb  1 21:22:46.069 EST
Policy        Segment         Outgoing    Outgoing            Next Hop        Bytes
Name          List            Label       Interface                           Switched
```

```
------------- --------------- ----------- ------------------- --------------- ------------
1013          PE1-PE3-PE2_link
                              15302       Te0/0/1/6           13.1.1.2        0
                 Label Stack (Top -> Bottom): { 15302 }
                 Path-id: 1, Weight: 0
                 Packets Switched: 0
   Local label: 24005
   Packets/Bytes Switched: 0/0
   (!): FRR pure backup

PE1#show mpls forwarding tunnels sr-policy name 1013
Thu Feb  1 21:23:22.743 EST
Tunnel        Outgoing    Outgoing     Next Hop        Bytes
Name          Label       Interface                    Switched
------------- ----------- ------------ --------------- ------------
1013         (SR) 15302       Te0/0/1/6    13.1.1.2        0

PE1#show l2vpn bridge-domain bd-name vpls501 detail
Sat Feb  3 11:27:35.655 EST
Legend: pp = Partially Programmed.
Bridge group: bg1, bridge-domain: vpls501, id: 250, state: up, ShgId: 0, MSTi: 0
……….
List of VFIs:
    VFI vpls1 (up)
      PW: neighbor 2.2.2.2, PW ID 501, state is up ( established )
        PW class pw100, XC ID 0xa00020d5
        Encapsulation MPLS, protocol LDP
        Source address 1.1.1.1
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set
        Preferred path Active : SR TE 100, Statically configured, fallback disabled
        Tunnel : Up
        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ -------------------------
          Label        41042                          24010
          Group ID     0xfa                           0x1
          Interface    vpls1                          vpls1
          MTU          1500                           1500
          Control word disabled                       disabled
          PW type      Ethernet                       Ethernet
          VCCV CV type 0x2                            0x2
                       (LSP ping verification)        (LSP ping verification)
          VCCV CC type 0x6                            0x6
                       (router alert label)           (router alert label)
                       (TTL expiry)                   (TTL expiry)
          ------------ ------------------------------ -------------------------
        Incoming Status (PW Status TLV):
          Status code: 0x0 (Up) in Notification message
        MIB cpwVcIndex: 2684362965
        Create time: 02/02/2018 16:20:59 (19:06:37 ago)

PE1# show l2vpn bridge-domain bd-name pwhe1 detail

Legend: pp = Partially Programmed.
Bridge group: bg3, bridge-domain: pwhe1, id: 3448, state: up, ShgId: 0, MSTi: 0
……
List of ACs:
    AC: PW-Ether1.1, state is up
List of Access PWs:
  List of VFIs:
    VFI pwhe1 (up)
      PW: neighbor 2.2.2.2, PW ID 600001, state is up ( established )
        PW class pw1000, XC ID 0xa0004401
        Encapsulation MPLS, protocol LDP
        Source address 1.1.1.1
        PW type Ethernet, control word disabled, interworking none
        Sequencing not set
        Preferred path Active : SR TE 1000, Statically configured, fallback disabled
        Tunnel : Up

        PW Status TLV in use
          MPLS         Local                          Remote
          ------------ ------------------------------ -------------------------
          Label        43292                          35464
          Group ID     0xd78                          0xd79
          Interface    pwhe1                          pwhe1
          MTU          1500                           1500
          Control word disabled                       disabled
          PW type      Ethernet                       Ethernet
          VCCV CV type 0x2                            0x2
                       (LSP ping verification)        (LSP ping verification)
          VCCV CC type 0x6                            0x6
                       (router alert label)           (router alert label)
                       (TTL expiry)                   (TTL expiry)
          ------------ ------------------------------ -------------------------
```

```
PE1# show l2vpn xconnect group xg2 xc-name pwhe10001 detail

Group xg2, XC pwhe10001, state is up; Interworking none
  AC: PW-Ether10001, state is up
    Type PW-Ether
    Interface-list: pwhe2
    Replicate status:
    Te0/0/1/9: success
    BE121: success
    Hu0/1/0/3: success
    MTU 1500; interworking none
    Internal label: 33018
     ……
PW: neighbor 2.2.2.2, PW ID 10001, state is up ( established )
    PW class pw1013, XC ID 0xa0000007
    Encapsulation MPLS, protocol LDP
    Source address 1.1.1.1
    PW type Ethernet, control word disabled, interworking none
    PW backup disable delay 0 sec
    Sequencing not set
    Preferred path Active : SR TE 1013, Statically configured, fallback disabled
    Tunnel : Up

    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        55510                          25363
      Group ID     0x203f1e0                      0x880
      Interface    PW-Ether10001                  TenGigE0/0/0/0/6.10001
      MTU          1500                           1500
      Control word disabled                       disabled
```

Associated Commands

- adjacency-sid
- index

- prefix-sid
- router isis
- segment-routing

The applicable segment routing commands are described in the *Segment Routing Command Reference for Cisco ASR 9000 Series Routers*

Related Topics

- Overview of Segment Routing , on page 769
- How Segment Routing Works , on page 770
- Segment Routing Global Block , on page 771

EVPN VPWS On-Demand Next Hop with SR-TE

The EVPN VPWS On-Demand Next Hop with SR-TE feature enables you to fetch the best path to send traffic from the source to destination in a point-to-point service using IOS XR Traffic Controller (XTC). On-Demand Next Hop (ODN) with SR-TE is supported on EVPN Virtual Private Wire Service (VPWS) and VPWS with pseudowire headend (PWHE).

When redistributing routing information across domains, provisioning of multi-domain services (Layer2 VPN and Layer 3 VPN) poses complexity and scalability issues. ODN with SR-TE feature delegates computation of an end-to-end Label Switched Path (LSP) to a path computation element (PCE). This PCE includes constraints and policies without any redistribution. It then installs the reapplied multi-domain LSP for the duration of the service into the local forwarding information base(FIB).

ODN uses BGP dynamic SR-TE capabilities and adds the path to the PCE. The PCE has the ability to find and download the end-to-end path based on the requirements. ODN triggers an SR-TE auto-tunnel based on the defined BGP policy. The PCE learns real-time topologies through BGP and/or IGP.

IOS XR Traffic Controller (XTC)

The path computation element (PCE) describes a set of procedures by which a path computation client (PCC) reports and delegates control of head-end tunnels sourced from the PCC to a PCE peer. The PCE peer requests the PCC to update and modify parameters of LSPs it controls. It also enables a PCC to allow the PCE to initiate computations and to perform network-wide orchestration.

Restrictions

- Maximum number of auto-provisioned TE policies is 1000.
- L2VPN services over PWHE with EVPN VPWS On-Demand Next Hop with SR-TE is not supported.
- PWHE over SR-TE with ECMP paths is not supported.

Configure EVPN VPWS On Demand Next Hop with SR-TE

Perform the following steps to configure EVPN VPWS On Demand Next Hop with SR-TE. The following figure is used as a reference to explain the configuration steps:

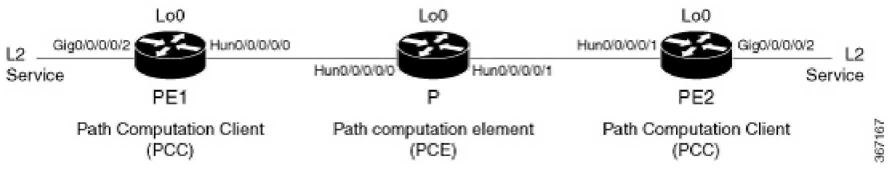
- Configure Prefix-SID in ISIS

- Configure SR-TE
- Configure PCE and PCC
- Configure SR color
- Configure EVPN route policy
- Configure BGP
- Configure EVPN VPWS
- Configure VPWS with pseudowire headend

Topology

Consider a topology where EVPN VPWS is configured on PE1 and PE2. Traffic is sent from PE1 to PE2 using SR-TE in the core. The PCE, which is configured on the P router, calculates the best path from PE1 to PE2. Path computation client (PCC) is configured on PE1 and PE2.

Figure 106: EVPN VPWS On Demand Next Hop with SR-TE



Configuration Example

Configure Prefix-SID in ISIS

Configure Prefix-SID in ISIS and topology-independent loop-free alternate path (TI-LFA) in the core such that each router uses a unique segment identifier associated with the prefix.

```
/* Configure Prefix-SID in ISIS and TI-LFA  on PE1 */

Router# configure
Route(config)# router isis ring
Route(config-isis)# is-type level-2-only
Route(config-isis)# net 49.0001.1921.6800.1001.00
Route(config-isis)# segment-routing global-block 30100 39100
Route(config-isis)# nsr
Route(config-isis)# distribute link-state
Route(config-isis)# nsf cisco
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# mpls traffic-eng level-1
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 30101
Route(config-isis-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/0
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# commit

/*Configure Prefix-SID in ISIS and TI-LFA  on P router */

Router# configure
Route(config)# router isis ring
Route(config-isis)# net 49.0001.1921.6800.1002.00
Route(config-isis)# segment-routing global-block 30100 39100
Route(config-isis)# nsr
Route(config-isis)# distribute link-state
Route(config-isis)# nsf cisco
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# mpls traffic-eng level-1
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 30102
Route(config-isis-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/0
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/1
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# commit

/* Configure Prefix-SID in ISIS and TI-LFA  on PE2 */

Router# configure
Route(config)# router isis ring
Route(config-isis)# net 49.0001.1921.6800.1003.00
Route(config-isis)# segment-routing global-block 30100 39100
Route(config-isis)# nsr
Route(config-isis)# distribute link-state
Route(config-isis)# nsf cisco
Route(config-isis)# address-family ipv4 unicast
Route(config-isis-af)# metric-style wide
Route(config-isis-af)# mpls traffic-eng level-1
Route(config-isis-af)# mpls traffic-eng router-id loopback0
Route(config-isis-af)# segment-routing mpls
Route(config-isis-af)# exit
!
Route(config-isis)# interface loopback0
Route(config-isis-if)# address-family ipv4 unicast
Route(config-isis-af)# prefix-sid index 30103
Route(config-isis-af)# exit
!
Route(config-isis)# interface HundredGigE0/0/0/1
Route(config-isis-if)# circuit-type level-1
Route(config-isis-if)# point-to-point
Route(config-isis-if)# hello-padding disable
Route(config-isis-if)# fast-reroute per-prefix
Route(config-isis-if-af)# fast-reroute per-prefix ti-lfa
Route(config-isis-if-af)# commit
```

Configure SR-TE

Configure SR-TE for P and PE routers.

```
/Configure SR-TE on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 1
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit
!
Router(config-sr-te)# on-demand color 2
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit
!
Router(config-sr-te)# on-demand color 3
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# commit

/*Configure SR-TE on P router */
Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# commit


/Configure SR-TE on PE2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# on-demand color 11
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit
!
Router(config-sr-te)# on-demand color 12
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# exit
!
Router(config-sr-te)# on-demand color 13
Router(config-sr-te-color)# dynamic
Router(config-sr-te-color-dyn-mpls)# pce
Router(config-sr-te-color-dyn-mpls-pce)# commit
```

Configure PCE and PCC

Configure PCE on P router, and PCC on PE1 and PE2. Optionally, you can configure multiple PCEs as well.

```
/* Configure PCC on PE1 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 205.1.0.1
Router(config-sr-te-pcc)# pce address ipv4 205.2.0.2
Router(config-sr-te-pcc)# commit

/* Configure PCE on P router */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# exit
Router(config)# pce
Router(config-pce)# address ipv4 205.2.0.2
Router(config-pce)# commit

/* Configure PCC on PE2 */

Router# configure
Router(config)# segment-routing
Router(config-sr)# traffic-eng
Router(config-sr-te)# pcc
Router(config-sr-te-pcc)# source-address ipv4 205.3.0.3
Router(config-sr-te-pcc)# pce address ipv4 205.2.0.2
Router(config-sr-te-pcc)# commit
```

Configure SR Color

Configure SR colors on PE routers.

```
/* Define SR color on PE1 */

Router# configure
Router(config)# extcommunity-set opaque color1
Router(config-ext)# 1
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color2
Router(config-ext)# 2
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color3
Router(config-ext)# 3
Router(config-ext)# end-set
!
/* Define SR color on PE2 */

Router# configure
Router(config)# extcommunity-set opaque color11
Router(config-ext)# 11
Router(config-ext)# end-set
!
```

```
Router(config)# extcommunity-set opaque color12
Router(config-ext)# 12
Router(config-ext)# end-set
!
Router(config)# extcommunity-set opaque color13
Router(config-ext)# 13
Router(config-ext)# end-set
!
```

Configure EVPN Route Policy

Configure EVPN route policy on PE1 and PE2. This example shows how to define the route policy language and track the EVPN route. The "rd" refers to the address of the PE and acts as Ethernet virtual interconnect for the L2 service.

```
/* Configure EVPN route policy on PE1 */

Router# configure
Router(config)# route-policy evpn_odn_policy
Router(config-rpl)# if rd in (205.3.0.3:2) then
Router(config-rpl-if)# set extcommunity color color1
Router(config-rpl-if)# set next-hop 205.3.0.3
Router(config-rpl-if)# elseif rd in (205.3.0.3:3) then
Router(config-rpl-elseif)# set extcommunity color color2
Router(config-rpl-elseif)# set next-hop 205.3.0.3
Router(config-rpl-elseif)# elseif rd in (205.3.0.3:4) then
Router(config-rpl-elseif)# set extcommunity color color3
Router(config-rpl-elseif)# set next-hop 205.3.0.3
Router(config-rpl-elseif)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy


/* Configure EVPN route policy on PE2 */

Router# configure
Router(config)# route-policy evpn_odn_policy
Router(config-rpl)# if rd in (205.1.0.1:2) then
Router(config-rpl-if)# set extcommunity color color11
Router(config-rpl-if)# set next-hop 205.1.0.1
Router(config-rpl-if)# elseif rd in (205.1.0.1:3) then
Router(config-rpl-elseif)# set extcommunity color color12
Router(config-rpl-elseif)# set next-hop 205.1.0.1
Router(config-rpl-elseif)# elseif rd in (205.1.0.1:4) then
Router(config-rpl-elseif)# set extcommunity color color13
Router(config-rpl-elseif)# set next-hop 205.1.0.1
Router(config-rpl-elseif)# endif
Router(config-rpl)# pass
Router(config-rpl)# end-policy
```

Configure BGP

Configure BGP on PE1 and PE2.

```
/* Configure BGP on PE1 */

Router# configure
Router(config)# router bgp 100
Routerconfig-bgp)# bgp router-id 205.1.0.1
Routerconfig-bgp)# bgp graceful-restart
```

```
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
!
Router(config-bgp)# neighbor 205.3.0.3
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy evpn_odn_policy in
Router(config-rpl)# commit

/* Configure BGP on PE2 */

Router# configure
Router(config)# router bgp 100
Routerconfig-bgp)# bgp router-id 205.3.0.3
Routerconfig-bgp)# bgp graceful-restart
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
!
Router(config-bgp)# neighbor 205.1.0.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback 0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# route-policy evpn_odn_policy in
Router(config-rpl)# commit
```

Configure EVPN VPWS

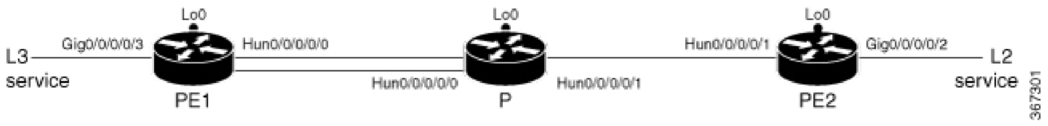
Configure EVPN VPWS on PE1 and PE2.

```
/* Configure EVPN VPWS on PE1 */

Router# configure
Router(config)# interface GigE0/0/0/2.2 l2transport
Router(config-subif)# encapsulation dot1q 1
Router# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e1_10
Router(config-l2vpn-xc-p2p)# interface GigE0/0/0/2.2
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 10 source 10
Router(config-l2vpn-xc-p2p)#commit

/* Configure EVPN VPWS on PE2 */

Router# configure
Router(config)# interface GigE0/0/0/2.4 l2transport
Router(config-subif)# encapsulation dot1q 1
Router# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e3_30
Router(config-l2vpn-xc-p2p)# interface GigE0/0/0/2.4
Router(config-l2vpn-xc-p2p)# neighbor evpn evi 2 target 10 source 10
Router(config-l2vpn-xc-p2p)#commit
```

Configure VPWS with Pseudowire Headend

Configure VPWS with pseudowire headend. The L3 services are used on the customer side with PWHE.

Figure 107: VPWS with Pseudowire Headend



```
Router# configure
Router(config)# generic-interface-list ifl
Router(config-gen-if-list)# interface HundredGigE0/0/0/0
Router(config-gen-if-list)# exit
!
Router(config)# interface PW-Ether18
Router(config-if)# attach generic-interface-list ifl
Router(config-gen-if-list)# exit
!
Router# configure
Router(config)# interface PW-Ether18.4
Router(config-if)# ipv4 address 123.123.123.1 255.255.255.0
Router(config-if)# encapsulation dot1q 4
Router(config-if)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e3_30
Router(config-l2vpn-xc-p2p-pw)# interface PW-Ether18
Router(config-l2vpn-xc-p2p-pw)# neighbor evpn evi 4 target 30 source 30
Router(config-l2vpn-xc-p2p-pw)# exit
!
Router# configure
Router(config)# interface GigE0/0/0/2.4 l2transport
Router(config-if)# encapsulation dot1q 4
Router(config-if)# exit
!
Router(config)# l2vpn
Router(config-l2vpn)# xconnect group evpn_vpws
Router(config-l2vpn-xc)# p2p e3_30
Router(config-l2vpn-xc-p2p)# interface GigE0/0/0/2.4
Router(config-l2vpn-xc-p2p-pw)# neighbor evpn evi 4 target 30 source 30
Router(config-l2vpn-xc-p2p-pw)# commit
```

Running Configuration

```
/* Configure Prefix-SID in ISIS and TI-LFA */

PE1:

configure
 router isis ring
  net 49.0001.1921.6800.1001.00
  segment-routing global-block 30100 39100
  nsr
  distribute link-state
  nsf cisco
  address-family ipv4 unicast
```

```
   metric-style wide
   mpls traffic-eng level-1
   mpls traffic-eng router-id Loopback0
   segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 30101
  !
 !
 interface HundredGigE0/0/0/0
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
 !
```

P:

```
configure
 router isis ring
  net 49.0001.1921.6800.1002.00
  segment-routing global-block 30100 39100
  nsr
  distribute link-state
  nsf cisco
  address-family ipv4 unicast
   metric-style wide
   mpls traffic-eng level-1
   mpls traffic-eng router-id Loopback0
   segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 30102
  !
 !
 interface HundredGigE0/0/0/0
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
 !
 interface HundredGigE0/0/0/1
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
```

PE2:

```
configure
 router isis ring
  net 49.0001.1921.6800.1003.00
  segment-routing global-block 30100 39100
  nsr
  distribute link-state
  nsf cisco
  address-family ipv4 unicast
   metric-style wide
   mpls traffic-eng level-1
   mpls traffic-eng router-id Loopback0
   segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 30103
  !
 !
 interface HundredGigE0/0/0/1
  circuit-type level-1
  point-to-point
  hello-padding disable
  address-family ipv4 unicast
   fast-reroute per-prefix
   fast-reroute per-prefix ti-lfa
  !
 !


/* Configure SR-TE */

PE1:

configure
 segment-routing
  traffic-eng
   on-demand color 1
    dynamic
     pce
   !
  !
  on-demand color 2
   dynamic
    pce
    !
   !
  !
  on-demand color 3
   dynamic
    pce
    !
P:

configure
 segment-routing
  traffic-eng
!

PE2:

configure
 segment-routing
  traffic-eng
   on-demand color 11
    dynamic
     pce
   !
  !
  on-demand color 12
   dynamic
    pce
    !
   !
  !
  on-demand color 13
   dynamic
    pce
    !

/* Configure PCE and PCC */

PE1:

configure
 segment-routing
  traffic-eng
   pcc
    source-address ipv4 205.1.0.1
    pce address ipv4 205.2.0.2
    !
P:

configure
 segment-routing
  traffic-eng
 pce
  address ipv4 205.2.0.2
    !

PE2:

configure
 segment-routing
  traffic-eng
   pcc
    source-address ipv4 205.3.0.3
    pce address ipv4 205.2.0.2
    !

/* Configure SR Color */

PE1:

configure
 extcommunity-set opaque color1
  1
end-set
!
 extcommunity-set opaque color2
  2
end-set
!
 extcommunity-set opaque color3
  3
end-set
!

PE2:

configure
 extcommunity-set opaque color11
  11
```

```
end-set
!
 extcommunity-set opaque color12
  12
end-set
!
 extcommunity-set opaque color13
  13
end-set
!


/* Configure EVPN route policy */

PE1:

configure
 route-policy evpn_odn_policy
  if rd in (205.3.0.3:2) then
    set extcommunity color color1
    set next-hop 205.3.0.3
  elseif rd in (205.3.0.3:3) then
    set extcommunity color color2
    set next-hop 205.3.0.3
  elseif rd in (205.3.0.3:4) then
    set extcommunity color color3
    set next-hop 205.3.0.3
  endif
 pass
end-policy

PE2:

configure
 route-policy evpn_odn_policy
  if rd in (205.1.0.1:2) then
    set extcommunity color color11
    set next-hop 205.1.0.1
  elseif rd in (205.1.0.1:3) then
    set extcommunity color color12
    set next-hop 205.1.0.1
  elseif rd in (205.1.0.1:4) then
    set extcommunity color color13
    set next-hop 205.1.0.1
  endif
 pass
end-policy

/* Configure BGP */

PE1:

configure
 router bgp 100
  bgp router-id 205.1.0.1
  bgp graceful-restart
  address-family l2vpn evpn
 !
 neighbor 205.3.0.3
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
   route-policy evpn_odn_policy in
  !
```

```
PE2:

configure
 router bgp 100
  bgp router-id 205.3.0.3
  bgp graceful-restart
  address-family l2vpn evpn
 !
 neighbor 205.1.0.1
  remote-as 100
  update-source Loopback0
  address-family l2vpn evpn
   route-policy evpn_odn_policy in
  !

/* Configure EVPN VPWS */

PE1:

configure
 interface GigE0/0/0/2.2 l2transport
  encapsulation dot1q 1
!
l2vpn
 xconnect group evpn_vpws
  p2p e1_10
   interface GigE0/0/0/2.2
   neighbor evpn evi 2 target 10 source 10
  !
 !

PE2:

configure
 interface GigE0/0/0/2.4 l2transport
  encapsulation dot1q 1
!
l2vpn
 xconnect group evpn_vpws
  p2p e3_30
   interface GigE0/0/0/2.4
   neighbor evpn evi 2 target 10 source 10
  !
 !
!

/* Configure VPWS with Pseudowire Headend */

generic-interface-list ifl
 interface HundredGigE0/0/0/0
!
interface PW-Ether18
 attach generic-interface-list ifl
!
interface PW-Ether18.4
 ipv4 address 123.123.123.1 255.255.255.0
encapsulation dot1q 4
!
l2vpn
 xconnect group evpn_vpws
  p2p e3_30
   interface PW-Ether18
   neighbor evpn evi 4 target 30 source 30
  !
interface GigE0/0/0/2.4 l2transport
 encapsulation dot1q 4
!
l2vpn
 xconnect group evpn_vpws
  p2p e3_30
   interface GigE0/0/0/2.4
   neighbor evpn evi 4 target 30 source 30
  !
```

Verify EVPN VPWS On Demand Next Hop with SR-TE Configuration

Verify if SR-TE policy is auto-provisioned for each L2 service configured on EVPN ODN.

```
PE1# show segment-routing traffic-eng policy

SR-TE policy database
---------------------

Name: bgp_AP_1 (Color: 1, End-point: 205.3.0.3)
    Status:
      Admin: up  Operational: up for 07:16:59 (since Oct  3 16:47:04.541)
    Candidate-paths:
      Preference 100:
        Dynamic (pce 205.2.0.2) (active)
          Weight: 0
            30103 [Prefix-SID, 205.3.0.3]
    Attributes:
      Binding SID: 68007
        Allocation mode: dynamic
        State: Programmed
        Policy selected: yes
      Forward Class: 0
      Distinguisher: 0
    Auto-policy info:
      Creator: BGP
      IPv6 caps enable: no
PE1#show l2vpn xconnect group evpn_vpws xc-name evpn_vpws_1001 detail
Group evpn_vpws, XC evpn_vpws_1001, state is up; Interworking none
  AC: Bundle-Ether12.1001, state is up
    Type VLAN; Num Ranges: 1
    Outer Tag: 1000
    Rewrite Tags: []
    VLAN ranges: [1, 1]
    MTU 1500; XC ID 0xc0000018; interworking none
    Statistics:
      packets: received 642304, sent 642244
      bytes: received 61661184, sent 61655424
      drops: illegal VLAN 0, illegal length 0
  EVPN: neighbor 20.20.20.20, PW ID: evi 1001, ac-id 10001, state is up ( established )
    XC ID 0xa0000007
    Encapsulation MPLS
    Source address 10.10.10.10
    Encap type Ethernet, control word enabled
    Sequencing not set
    Preferred path Active : SR TE pref_sid_to_PE3, On-Demand, fallback enabled
    Tunnel : Up
    Load Balance Hashing: src-dst-mac

PE1#show bgp l2vpn evpn route-type 1
```

```
BGP router identifier 205.1.0.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0 RD version: 0
BGP main routing table version 36
BGP NSR Initial initsync version 25 (Reached)
BGP NSR/ISSU Sync-Group versions 36/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Network Next Hop Metric LocPrf Weight Path
Route Distinguisher: 205.1.0.1:2 (default for vrf VPWS:2)
*>i[1][0000.0000.0000.0000.0000][1]/120
205.3.0.3 T:bgp_AP_1
100 0 i

PE1# show evpn evi ead detail

EVI Ethernet Segment Id EtherTag Nexthop Label SRTE IFH
----- ------------------------ -------- --------------------------------------- --------
----------
2 0000.0000.0000.0000.0000 1 205.3.0.3 24000 0x5a0
Source: Remote, MPLS
```

Associated Commands

- adjacency-sid
- index
- prefix-sid
- router isis
- segment-routing

The applicable segment routing commands are described in the *Segment Routing Command Reference for Cisco ASR 9000 Series Routers*

Related Topics

- Overview of Segment Routing , on page 769
- How Segment Routing Works , on page 770
- Segment Routing Global Block , on page 771