Preface

The *Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router* preface contains the following sections:

- Changes to This Document, page VPC-xv
- Obtaining Documentation and Submitting a Service Request, page VPC-xv

Changes to This Document

Table 1 lists the technical changes made to this document since it was first printed.

Table 1 Changes to This Document

Revision	Date	Change Summary
OL-28400-01	December 2012	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.

New and Changed Information in Release 4.3.x

This table summarizes the new and changed feature information for the Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco XR 12000 Series Router, and tells you where they are documented.

Table 2 New and Changed Features

Feature	Description	Introduced/Changed in Release	Where Documented
Flow Aware Transport Pseudowire (FAT PW)	This feature was introduced.	Release 4.3.0	*Implementing Virtual Private LAN Services module* • Flow Aware Transport Pseudowire (FAT PW) Overview • Configuring Flow Aware Transport Pseudowire Refer *Virtual Private Network Commands* in *Cisco IOS XR Virtual Private Network Command Reference for the Cisco XR 12000 Series Router, Release 4.3.x* for information on the commands used for configuring and verifying Flow Aware Transport Pseudowire (FAT PW) feature.
L2VPN Nonstop Routing	This feature was introduced.	Release 4.3.0	*Implementing MPLS Layer 2 VPNs* • L2VPN Nonstop Routing • Configuring L2VPN Nonstop Routing Refer *Virtual Private Network Commands* in *Cisco IOS XR Virtual Private Network Command Reference for the Cisco XR 12000 Series Router, Release 4.3.x* for information on the commands used for configuring and verifying L2VPN Nonstop Routing feature.
Pseudowire Grouping	This feature was introduced.	Release 4.3.0	*Implementing Virtual Private LAN Services* • Pseudowire Grouping • Enabling Pseudowire Grouping

Implementing MPLS Layer 2 VPNs

This module provides the conceptual and configuration information for MPLS Layer 2 virtual private networks (VPNs) on Cisco IOS XR software.

For the functionality of MPLS VPNs over IP Tunnels, see Implementing MPLS VPNs over IP Tunnels in *Cisco IOS XR Virtual Private Network Configuration Guide*.

Note For more information about MPLS Layer 2 VPN on the Cisco IOS XR software and for descriptions of the commands listed in this module, see the "Related Documents" section. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing MPLS Layer 2 VPN Configuration Module

Release	Modification
Release 3.4.0	This feature was introduced.
Release 3.4.1	Support was added for: • Virtual Circuit Connection Verification (VCCV) on L2VPN • QinQ mode and QinAny mode for EoMPLS
Release 3.5.0	Support was added for: • EoMPLS Inter-AS mode • Mac-in-Mac protocol
Release 3.6.0	Support was added for: • Ethernet Remote Port Shutdown
Release 3.7.0	Support was added for ATM over MPLS (ATMoMPLS) with Layer 2VPN capability.

Release 3.8.0	Support was added for Any Transport over MPLS (AToM) for: • IP Interworking on Engine 3 and 5 Line Cards • PPP/HDLC Like-to-Like Pseudowires on Engine 3 and Engine 5 Line Cards • ATM Like-to-Like Pseudowires on Engine 3 and Engine 5 Line Cards • Frame Relay DLCI, and MLFR Like-to-Like Pseudowires on Engine 3 Line Cards • Ethernet Port Mode and VLAN Like-to-Like Pseudowires on Engine 3 Line Cards • Local Switching Support with L2TPv3 on Engine 3 and Engine 5 Line Cards
Release 4.0.1	Support was added for the ATM Interworking feature.
Release 4.2.0	Support was added for Any Transport over MPLS (AToM) for: • IP Interworking support on cHDLC and PPP attachment circuits • FR-to-Ethernet bridged interworking • Local switching for PPP and cHDLC Support was added for Circuit Emulation (CEM) over Packet
Release 4.3.0	Support was added for these features: • L2VPN nonstop routing (NSR) • Pseudowire Grouping

Contents

Prerequisites for Implementing MPLS L2VPN

To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

If you need assistance with your task group assignment, contact your system administrator.

Information About Implementing L2VPN

To implement MPLS L2VPN, you should understand the following concepts:

- ATMoMPLS with L2VPN Capability, page VPC-17
- Virtual Circuit Connection Verification on L2VPN, page VPC-18
- Ethernet over MPLS, page VPC-19
- Quality of Service, page VPC-22
- High Availability, page VPC-23
- Preferred Tunnel Path, page VPC-23
- Any Transport over MPLS, page VPC-24
- Circuit Emulation Over Packet Switched Network, page VPC-31
- L2VPN Nonstop Routing, page VPC-32
- Pseudowire Grouping, page VPC-33

L2VPN Overview

Layer 2 VPN (L2VPN) emulates the behavior of a LAN across an IP or MPLS-enabled IP network allowing Ethernet devices to communicate with each other as they would when connected to a common LAN segment.

As Internet service providers (ISPs) look to replace Frame Relay or their Asynchronous Transfer Mode (ATM) infrastructures with an IP infrastructure, there is a need for to provide standard methods of using an IP infrastructure to provide a serviceable L2 interface to customers; specifically, to provide standard ways of using an IP infrastructure 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 the following 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.

ATMoMPLS with L2VPN Capability

These topics describe the ATM over MPLS (ATMoMPLS) with L2VPN feature:

- ATMoMPLS with L2VPN Overview, page VPC-18
- Layer 2 Local Switching Overview, page VPC-18
- ATM Adaptation Layer 5, page VPC-18

ATMoMPLS with L2VPN Overview

The ATMoMPLS feature supports ATM Adaptation Layer 5 (AAL5) transport. ATMoMPLS is a type of Layer 2 point-to-point connection over an MPLS core. ATMoMPLS and ATM local switching are supported only for ATM-to-ATM interface-to-interface switching combinations.

To implement the ATMoMPLS feature, the Cisco CRS-1 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 CRS-1 routers.

Layer 2 Local Switching Overview

Local switching lets you to switch Layer 2 data between two interfaces of the same type (for example, ATM-to-ATM, or Frame Relay-to-Frame Relay) or between interfaces of different types (for example, Frame Relay to ATM) on the same router, over an IP core network. The interfaces are on the same line card or on two different cards. During these types of switching, Layer 2 address is used instead of the Layer 3 address.

In addition, same-port local switching lets you to switch Layer 2 data between two circuits on the same interface.

ATM Adaptation Layer 5

AAL5 lets you transport AAL5 PDUs from various customers over an MPLS backbone. ATM AAL5 extends the usability of the MPLS backbone by enabling it to offer Layer 2 services in addition to already existing Layer 3 services. You can enable the MPLS backbone network to accept AAL5 PDUs by configuring the provider edge (PE) routers at both ends of the MPLS backbone.

To transport AAL5 PDUs over MPLS, a virtual circuit is set up from the ingress PE router to the egress PE router. This virtual circuit transports the AAL5 PDUs from one PE router to the other. Each AAL5 PDU is transported as a single packet.

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 IOS XR software 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 the following subsections:

- Ethernet Port Mode, page VPC-19
- VLAN Mode, page VPC-20
- VLAN Mode, page VPC-20
- Inter-AS Mode, page VPC-20
- QinQ Mode, page VPC-21
- QinAny Mode, page VPC-22
- Mac-in-Mac Protocol (Provide Backbone Bridging), page VPC-22

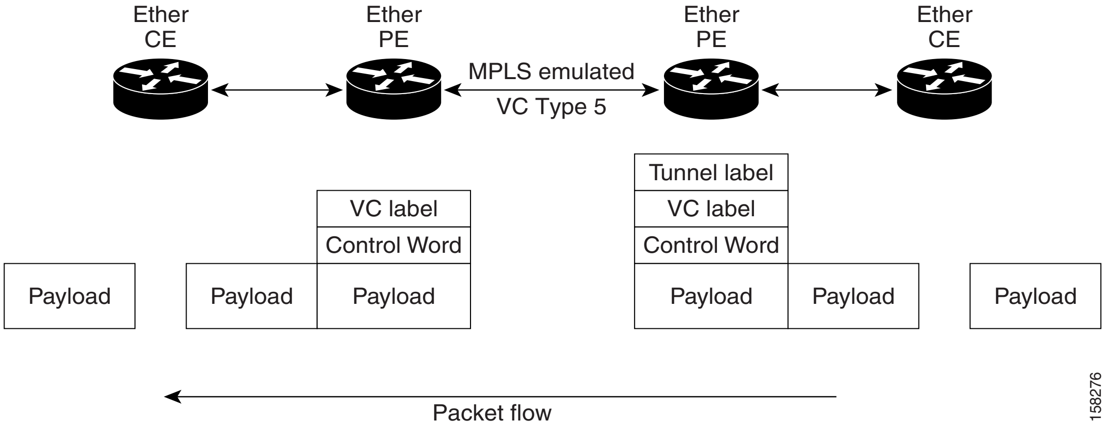
Ethernet Port Mode

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.

Note L2VPN forwarding using GRE tunnels is supported in the Ethernet port mode.

Figure 1 provides an example of Ethernet port mode.

Figure 1 Ethernet Port Mode Packet Flow

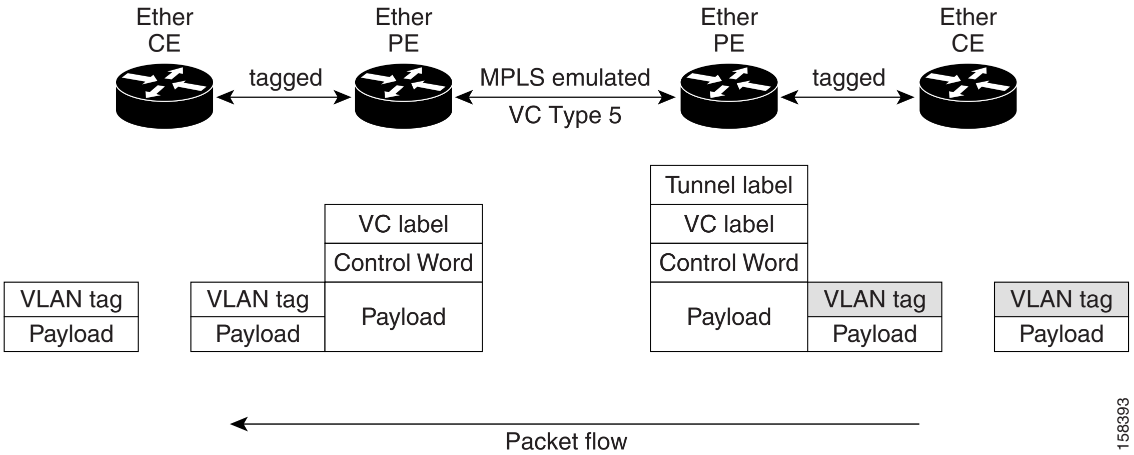


VLAN Mode

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 4 is the default mode.

As illustrated in Figure 2, 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 2 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.

Note L2VPN forwarding using GRE tunnels is supported in the VLAN mode.

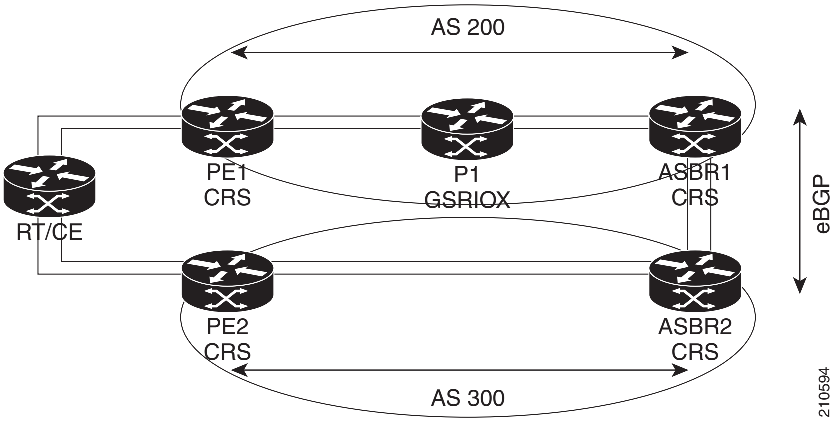
Inter-AS Mode

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.

Figure 3 illustrates MPLS over Inter-AS with a basic double AS topology with iBGP/LDP in each AS.

Figure 3 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 CRS-1 router implements 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:

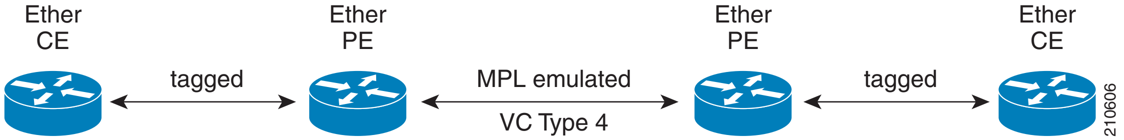
- IPv4 unicast and multicast
- IPv6 unicast and multicast
- MPLS
- Connectionless Network Service (CLNS) for use by Intermediate System-to-Intermediate System (IS-IS) Protocol

Note The Cisco CRS-1 router does not support: bundle attachment circuits and Hot Standby Router Protocol (HSRP) or Virtual Router Redundancy Protocol (VRRP) on QinQ subinterfaces.

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).

Figure 4 illustrates QinQ using VC type 4.

Figure 4 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.

Mac-in-Mac Protocol (Provide Backbone Bridging)

The Mac-in-Mac (or, Provider Backbone Bridging) protocol lets service providers scale networks using Ethernet technology to maintain management and operational simplicity, and reduce operating costs.

Mac-In-Mac encapsulates the customer MAC header with a service provider MAC header. Instead of using additional Q-tags to separate end customers, a 24-bit service tag in the service provider encapsulating MAC header is used, which provides support for up to 16-million service instances.

Note Mac-In-Mac is standardized as IEEE 802.1ah.

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*). Figure 5 illustrates L2 and L3 QoS service policies in a typical L2VPN network.

Figure 5 L2VPN QoS Feature Application

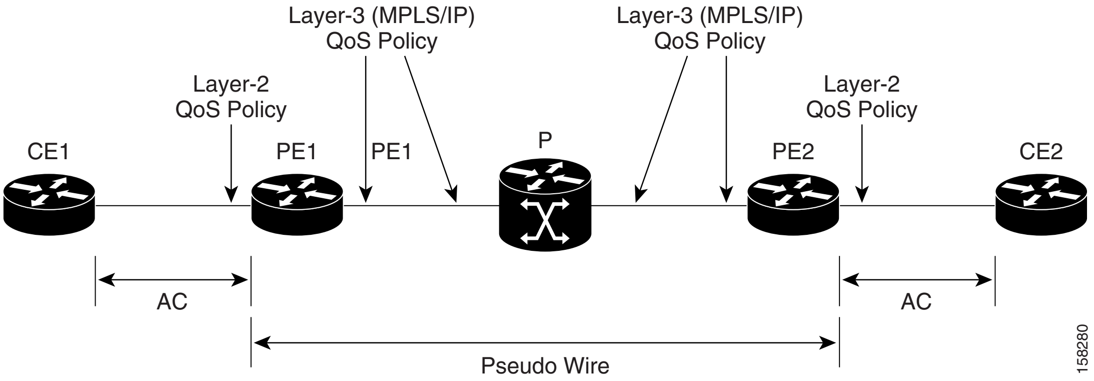
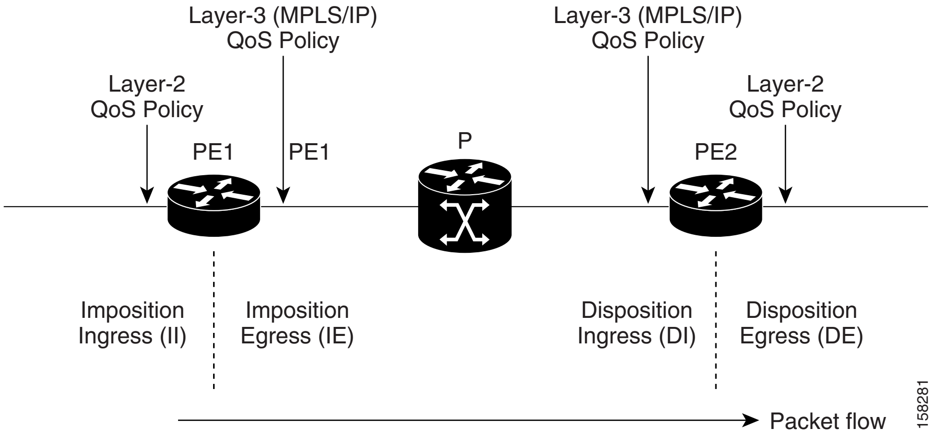


Figure 6 shows four packet processing paths within a provider edge device where a QoS service policy can be attached. In an L2VPN network, packets are received and transmitted on the edge-facing interfaces as L2 packets and transported on the core-facing interfaces as MPLS (EoMPLS) or IP (L2TP) packets.

Figure 6 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.

Note The l2tp_mgr process does not support high availability.

The availability of L2VPN meets the following 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.
- The fallback enable option is supported.

Any Transport over MPLS

Any Transport over MPLS (AToM) transports Layer 2 packets over a Multiprotocol Label Switching (MPLS) backbone, which 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 the connection, called a *pseudowire*, between the routers. You specify the following information on each PE router:

- The type of Layer 2 data that will be transported across the pseudowire, such as Ethernet, Frame Relay, or ATM
- 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

These topics describe the AToM feature:

- IP or Routed Interworking, page VPC-24
- Like-to-Like Pseudowires, page VPC-31
- Control Word Processing, page VPC-30

IP or Routed Interworking

In AToM IP Interworking, also called *routed interworking*, the carrier edge (CE) routers encapsulate IP on the link between the CE and PE routers. A new VC type is used to signal the IP pseudowire in MPLS and L2TPv3. Translation between the Layer 2 and IP encapsulations across the pseudowire is required.

IP Interworking is used to provide IP connectivity between sites, regardless of the Layer 2 connectivity to these sites. It is different from a Layer 3 VPN, because it is point-to-point in nature and the service provider does not maintain any customer routing information.

These modes support IP Interworking on AToM:

- ATM to Ethernet: In this interworking, both ATM and Ethernet PE routers are configured for IP interworking. IP packets from an ATM CE are encapsulated using IP over MPLS and transmitted over the pseudowire. On the Ethernet side, the Ethernet PE removes the Layer 2 framing on the Ethernet packets from the Ethernet CE and forwards the IP packet on the pseudowire using IP over MPLS encapsulation. Non-IP packets are dropped in this process. At the ATM PE, after label disposition, the IP packets are encapsulated over AAL5 using IP encapsulation. In either direction, packets for which translations are not supported, are dropped.
- Ethernet port to VLAN mode: Using the Ethernet port mode, you can create an Ethernet virtual local area network (VLAN) among geographically separated sites. Different sites can operate together over an MPLS network as though they were on a common Ethernet network.
- Frame Relay to Ethernet: Multi-protocol Frame Relay packets from the Frame Relay CE are encapsulated using IP over MPLS and transmitted over the pseudowire. On the Ethernet side, the Ethernet PE removes the Layer 2 framing on the Ethernet packets from the Ethernet CE and forwards

the Layer 3 packet over the pseudowire using IP over MPLS encapsulation. At the Frame Relay PE, after label disposition, the Layer 3 packets are encapsulated over Frame Relay using IP encapsulation. In either direction, packets for which translations are not supported are dropped.

- Frame Relay to ATM AAL5: ATM and Frame Relay links are locally terminated and IP interworking is used to transport the Layer 3 packets over the IP over MPLS pseudowire.
- ATM AAL5—ATM Adaptation Layer Type-5 (AAL5) allows efficient transportation of PVCs across the MPLS backbone. Multiple PVCs can be multiplexed onto a single label switched path between the provider edge routers.
- Point-to-Point—In this interworking, the point-to-point protocol (PPP) session is terminated at the PE while interworking with PPP attachment circuits. The PE router is responsible for negotiating LCP and IPCP with the CE router. PPP on the PE router can be configured with the **ppp ipcp address proxy ip-address** command where the remote CE router's IP address is used. This IP address is used by the PE router during IPCP negotiations with the CE router.
- Cisco High-Level Data Link Control (cHDLC)—Interworking with cHDLC attachment circuits works in the same way as interworking with PPP attachment circuits. However, *keepalive* messages are sent and received between the PE and CE routers to keep the L2VPN attachment circuit active.

These types of cross connections are supported for AToM IP Interworking:

- Ethernet
  - VLAN
  - Q-in-Q
  - Frame Relay
  - ATM AAL5 SNAP/MUX/NLPID
- VLAN
  - Ethernet
  - Q-in-Q
  - Frame Relay
  - ATM AAL5 SNALP/MUX/NLPID
- Q-in-Q
  - Ethernet
  - VLAN
  - Frame Relay
  - ATM AAL5 SNAP/MUX/NLPID
- Frame Relay
  - Ethernet
  - VLAN
  - Q-in-Q
  - ATM AAL5 SNAP/MUX/NLPID

ATM AAL5 to Ethernet Bridged Interworking

This interworking provides interoperability between ATM attachment virtual circuit (AC) and Ethernet attachment AC connected to different provider edge (PE) routers. The bridged encapsulation is used corresponding to the bridged (Ethernet) interworking mechanism.

The interworking function is performed at the PE connected to the ATM AC.

Processing at PE connected to ATM AC

In the direction from the ATM segment to MPLS cloud, the bridged encapsulation (ATM or SNAP header) is discarded and the ethernet frame is encapsulated with the labels required to pass through the pseudowire using the VC type 5 (Ethernet). ATM side is configured with encapsulation type as *aal5snap*. In the opposite direction, after the label disposition from the MPLS cloud, ethernet frames are encapsulated over AAL5 using bridged encapsulation.

These translations are supported:

- Ethernet without LAN FCS
- Spanning tree

The existing QoS functionality for ATM is supported, including setting the ATM CLP bit. Non-AAL5 traffic, (e.g. OAM cells) are processed at RP level. A VC that has been configured with OAM cell emulation on the ATM PE router (with **oam-ac emulation-enable** command) can send end-to-end F5 loopback cells at configured intervals toward the customer edge (CE) router. When the pseudowire is down, an F5 end-to-end segment alarm indication signal or remote defect indication (AIS/RD) is sent from the PE router to the CE router.

Restrictions

These restrictions must be considered:

- Only ATM AAL5 VC mode is supported. ATM VP and port mode are not supported.
- SVCs are not supported.

Processing at PE connected to Ethernet AC

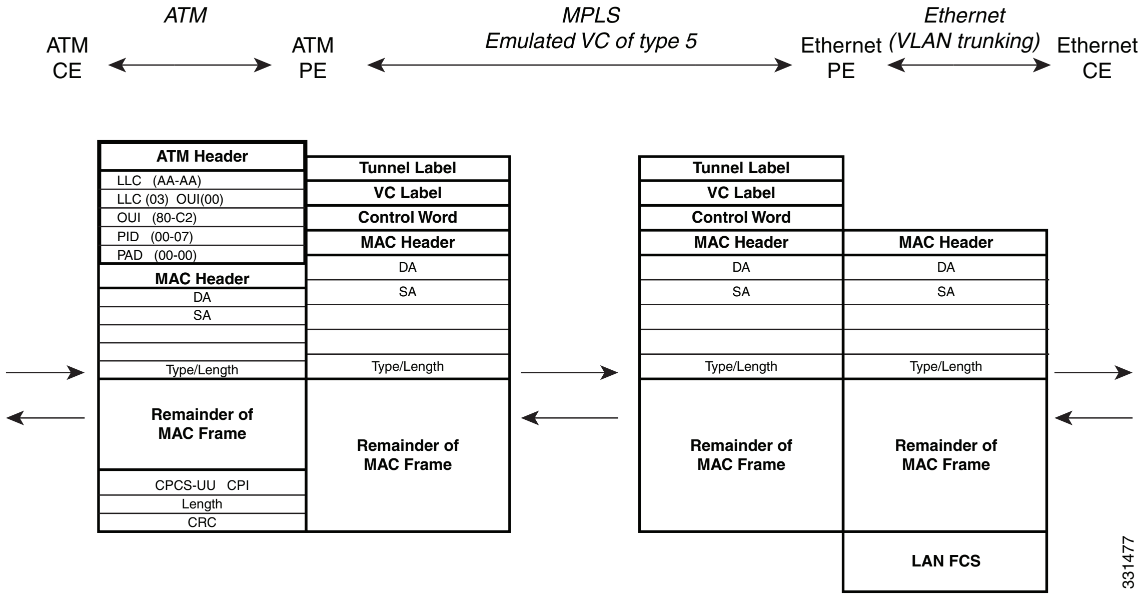
This section provides information on:

- Ethernet Port Mode
- Ethernet dot1q/qinq

Ethernet Port Mode

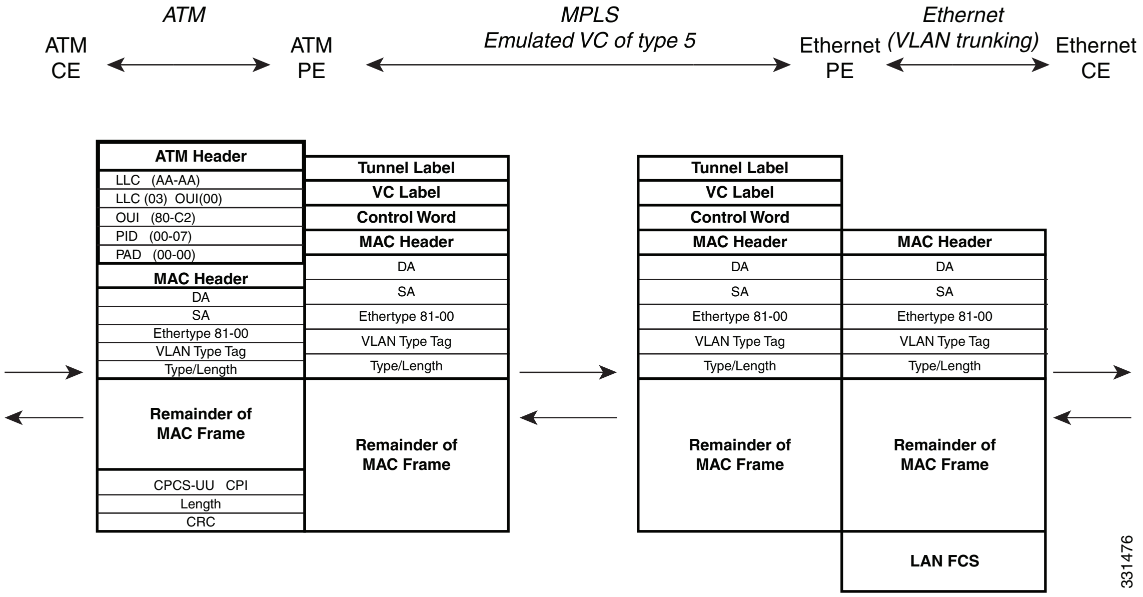
The Ethernet PE (connected to the Ethernet segment) operates similarly to Ethernet like-to-like services. For the packets coming from MPLS cloud, after the label disposition, the Ethernet frames are sent as is towards CE.

Figure 7 Protocol Stack for ATM to Ethernet AToM Bridged Interworking (without VLAN tag)



Note If the Ethernet frame arriving from Ethernet CE includes a 802.1Q header (VLAN header), due to the type of endpoint attachment (Ethernet port mode), the VLAN header stays in the frame across the pseudowire as shown in Figure 8.

Figure 8 Protocol Stack for ATM to Ethernet AToM Bridged Interworking (with Vlan tag)

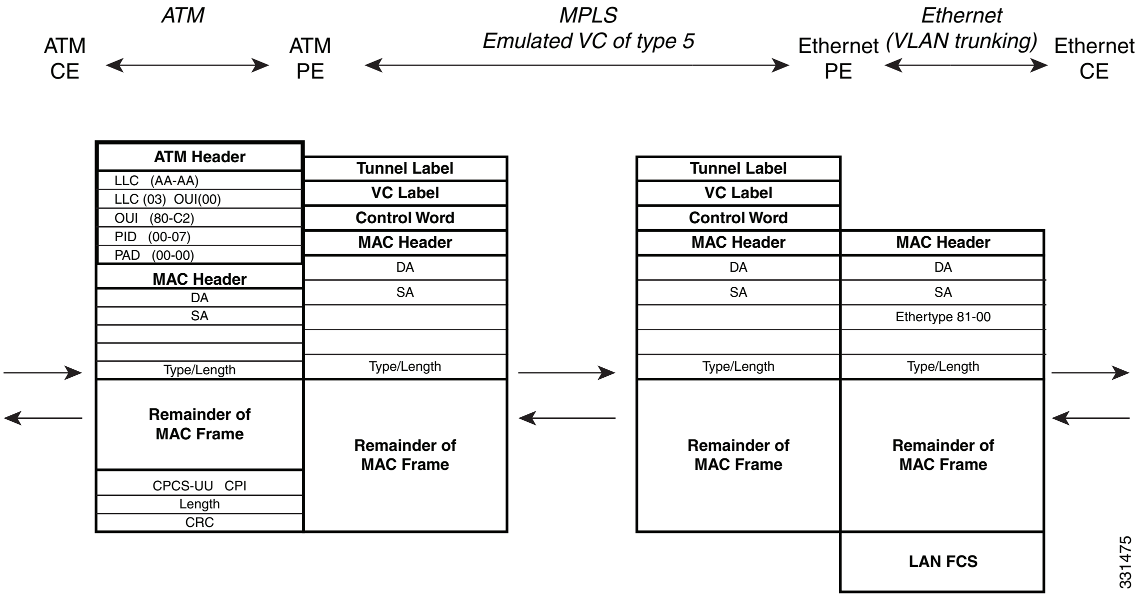


Ethernet dot1q/qinq

The PE connected to the Ethernet side discards the VLAN tags present in the incoming packets from the VLAN CE and pushed towards the MPLS cloud. For packets coming from MPLS cloud, it inserts VLAN tags into the Ethernet frames. Therefore, the frames sent on the pseudo wire (with VC type 5) are Ethernet frames without the VLAN header.

Note Ethernet frames received from the VLAN CE can contain more than two tags. Therefore, the number of tags processed or removed on the PE depends on the encapsulation type (dot1q/qinq) and the remaining tags are sent towards MPLS cloud as the payload.

Figure 9 Protocol Stack for ATM to VLAN AToM Bridged Interworking



Local Switching

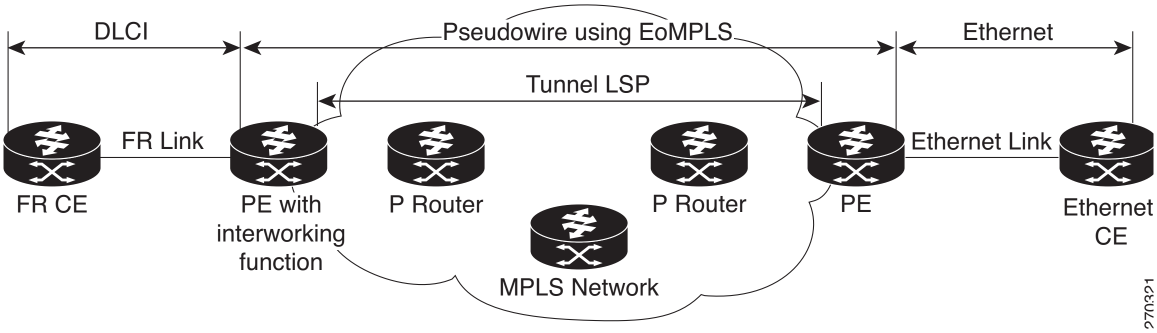
The functionality mentioned in the earlier sections applies to Local switching as well. The only difference is that, no PWE3 signaling is involved in bringing up the L2VPN circuit.

Ethernet or Bridged Interworking

Ethernet interworking is also called bridged interworking. Ethernet frames are bridged across the pseudowire. The CE routers could be natively bridging Ethernet or could be routing using a bridged encapsulation model. The PE routers operate in Ethernet like-to-like mode.

Figure 10 shows the reference network for Frame Relay (FR) to Ethernet bridged interworking.

Figure 10 Reference Network for Bridged Interworking



On the PE connected to FR attachment circuit (AC), in the direction from the FR segment to MPLS cloud, the Ethernet frames are received with the Frame Relay bridged encapsulation (FR/SNAP header). The SNAP header is discarded and the Ethernet frame is encapsulated with the labels required to pass through the pseudowire using the VC type 5 (Ethernet).

In the opposite direction, after the label disposition from the MPLS cloud, Ethernet frames are encapsulated over FR using bridged encapsulation.

Restrictions

These restrictions apply to the FR AC for the BRIW with Ethernet:

- At the FR AC, only these translations are supported and other translations are dropped:
  - Ethernet without LAN FCS (0300800080C20007)
  - Spanning tree (0300800080C2000E)
- The PVC status signaling works the same way as in the like-to-like case. The PE router reports the PVC status to the CE router based upon the availability of the pseudowire.
- The attachment circuit maximum transmission unit (MTU) must match when connected over MPLS.
- Only FR DLCI mode is supported. FR port mode is not supported.
- If the Ethernet frame includes a 802.1Q header (VLAN header), due to the type of endpoint attachment (Ethernet port mode), the VLAN header stays in the frame across the pseudowire.

The Ethernet PE (connected to the Ethernet segment) operates similarly to Ethernet like-to-like services. For the packets coming from MPLS cloud, after the label disposition, the Ethernet frames are sent as is towards the CE side.

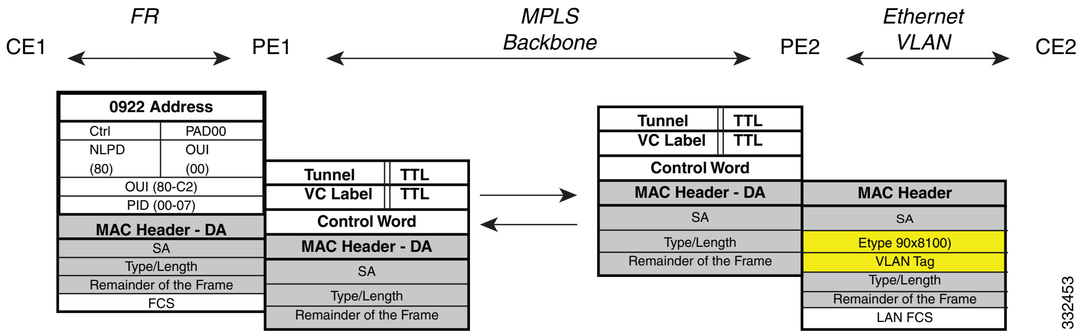
The PE connected to the Ethernet side, discards the VLAN tag(s) (Service Provider's) present in the incoming packets from the VLAN CE and pushes towards the MPLS cloud after adding the PWE3 Labels. For the packets coming from MPLS cloddish VLAN tag(s) are inserted into the Ethernet frames. Therefore, the frames sent on the pseudo wire (with VC type 5) are Ethernet frames without the Service Provider VLAN header.

Note Ethernet frames received from the VLAN CE or MPLS cloud can contain more than 2 tags. Therefore, the number of tags processed or removed on the PE depends on the type of encapsulation (dot1q/qinq) and the remaining tags are sent towards VLAN CE or MPLS cloud as the payload.

FR to Ethernet Local Switching

Figure 11 shows the local switching with bridged interworking.

Figure 11 Protocol Stack for FR to Ethernet(dot1Q/QinQ) Bridged Interworking



Local Switching with bridged interworking provides interoperability between Frame Relay attachment circuit and Ethernet attachment circuit connected to the same PE router. For this interworking type, bridged encapsulation is used corresponding to the bridged (Ethernet) interworking mechanism.

In the Ethernet to FR direction, the PE router forwards the Layer 2 packet without any change to the egress interface, encapsulating the L2 packet over FR using bridged encapsulation.

In the FR to Ethernet direction, the FR header and bridged encapsulation are discarded and the L2 packet is sent out with Ethernet encapsulation.

In local switching the only difference is that there is no PWE3 signaling involved in bringing up the L2VPN circuit.

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.

Like-to-Like Pseudowires

A pseudowire (PW) is a bidirectional VC connecting two Attached Circuits. In an MPLS network, PWs are carried inside an LSP tunnel.

A point-to-point (PPP) connection allows service providers to provide a transparent PPP pass-through where the customer-edge routers can exchange the traffic through an end-to-end PPP session. Service providers can offer a virtual leased-line solution, and use the PPP subinterface capability to peer with multiple providers through a single POS connection.

A High-Level Data Link control (HDLC) connection is emulated from a customer router to another customer router across an MPLS backbone. This technology allows transportation of HDLC frames across the packet networks. HDLC over MPLS also works in transparent mode.

Circuit Emulation Over Packet Switched Network

Circuit Emulation over Packet (CEoP) is a method of carrying Time Division Multiplexed (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 (Figure 12).

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)

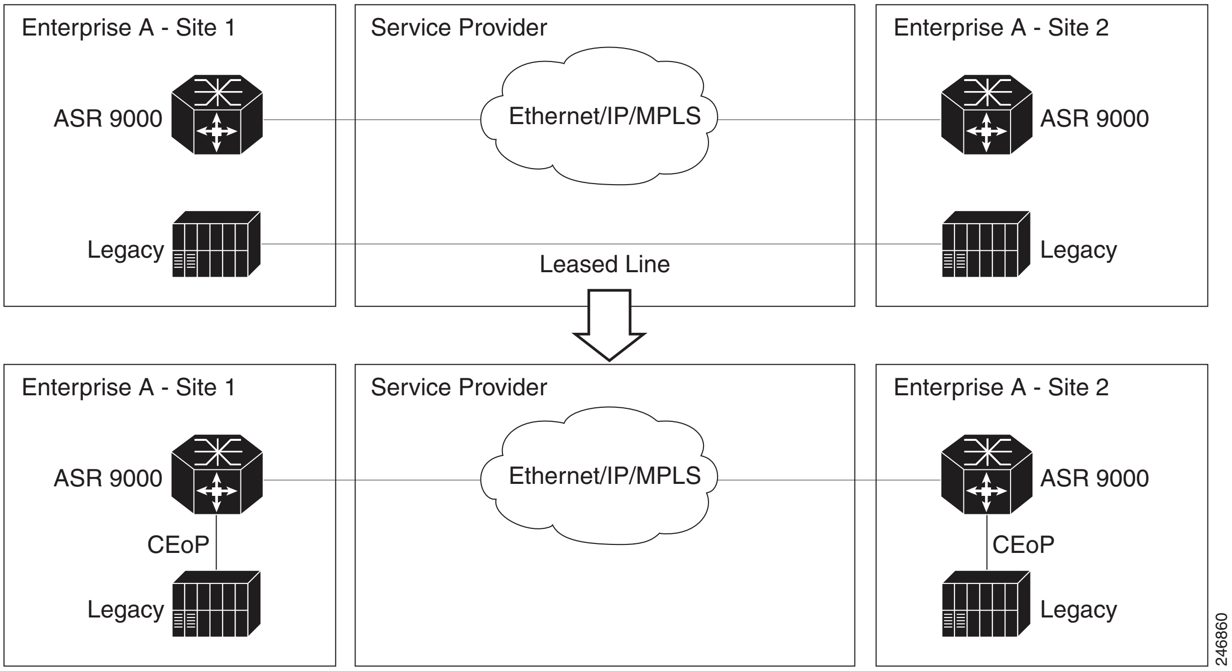
  Compared with SAToP, CESoPSN transmits emulated structured TDM signals. That is, it can identify and process the frame structure and transmit signaling in TDM frames. It may not transmit idle timeslot channels, but only extracts useful timeslots of CE devices from the E1 traffic stream and then encapsulates 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.

The CEM functionality is supported only on Cisco XR 12000 Series Router Engine 5 line cards having CEoP SPAs. CEM is supported on these variants of the CEoP SPAs:

- 24-Port Channelized T1/E1 ATM CEoP SPA (SPA-24CHT1-CE-ATM)
- 2-Port Channelized T3/E3 ATM CEoP SPA (SPA-2CHT3-CE-ATM)
- 1-port Channelized OC3 STM1 ATM CEoP SPA (SPA-1CHOC3-CE-ATM)

Figure 12 Enterprise Data Convergence using Circuit Emulation over Packet



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

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 (cras h) 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.

Pseudowire Grouping

When pseudowires (PW) are established, each PW is assigned a group ID that is common for all PWs created from the same physical port. Hence, when the physical port becomes non-functional or is deleted, L2VPN sends a single message to advertise the status change of all PWs belonging to the group. A single L2VPN signal thus avoids a lot of processing and loss in reactivity.

Note Pseudowire grouping is disabled by default.

How to Implement L2VPN

This section describes the tasks required to implement L2VPN:

- Configuring an Interface or Connection for L2VPN, page VPC-33
- Configuring Static Point-to-Point Cross-Connects, page VPC-35
- Configuring Dynamic Point-to-Point Cross-Connects, page VPC-37
- Configuring Inter-AS, page VPC-39
- Configuring L2VPN Quality of Service, page VPC-39
- Configuring Preferred Tunnel Path, page VPC-43
- Configuring AToM IP Interworking, page VPC-45
- Configuring Circuit Emulation Over Packet Switched Network, page VPC-64
- Configuring L2VPN Nonstop Routing, page VPC-72
- Enabling Pseudowire Grouping, page VPC-74

Configuring an Interface or Connection for L2VPN

Perform this task to configure an interface or a connection for L2VPN.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **l2transport**
4. **exit**
5. **interface** *type interface-path-id*
6. **dot1q native vlan** *vlan-id*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface GigabitEthernet 0/0/0/0`
		Enters interface configuration mode and configures an interface.

Step 3	**`l2transport`**
	Example:
	`RP/0/0/CPU0:router(config-if)# l2transport`
		Enables L2 transport on the selected interface.

Step 4	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-if-l2)# exit`
		Exits the current configuration mode.

Step 5	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface GigabitEthernet0/0/0/0`
		Enters interface configuration mode and configures an interface.

	Command or Action	Purpose
Step 6	**dot1q native vlan** *vlan ID* **Example:** `RP/0/0/CPU0:router(config-if)# dot1q vlan 1`	Assigns the native VLAN ID of a physical interface trunking 802.1Q VLAN traffic.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 Static Point-to-Point Cross-Connects

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

Please 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.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*

4. **p2p** *xconnect-name*
5. interworking ethernet
6. **interface** *type interface-path-id*
7. **neighbor** *ip-address* **pw-id** *pseudowire-id*
8. **mpls static label local** {*value*} **remote** {*value*}
9. **end**
   or
   commit
10. **show l2vpn xconnect group** *group name*

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enters L2VPN configuration mode.

Step 3	**`xconnect group`** *`group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group vlan_grp_1`
		Enters the name of the cross-connect group.

Step 4	**`p2p`** *`xconnect name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p vlan1`
		Enters a name for the point-to-point cross-connect.

Step 5	**`interworking ethernet`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# interworking ethernet`
		(Optional) Configures bridged interworking.

Step 6	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/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.

Step 7	**`neighbor`** *`ip-address`* **`pw-id`** *`pseudowire-id`*
	Example:
	`RP/0/0/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.

	Command or Action	Purpose
Step 8	**mpls static label local** {*value*} **remote** {*value*} **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# mpls static label local 699 remote 890`	Configures local and remote label ID values.

Step 9	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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.

Step 10	**show l2vpn xconnect group** *group name* **Example:** `RP/0/0/CPU0:show l2vpn xconnect group p2p`	Displays the name of the Point-to-Point cross-connect group you created.

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. To support MPLS Transport based PWs, configure the IGP Routing Protocol.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interworking ipv4 or interworking ethernet**
6. **interface** *type interface-path-id*

7. **neighbor** *ip-address* **pw-id** *pseudowire-id*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.
Step 3	**xconnect group** *group-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp_1`	Enters the name of the cross-connect group.
Step 4	**p2p** *xconnect-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p vlan1`	Enters a name for the point-to-point cross-connect.
Step 5	**interworking ipv4 or interworking ethernet** **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# interworking ipv4 or interworking ethernet`	Configures the interworking for IPv4 or Ethernet networks.
Step 6	**interface** *type interface-path-id* **Example:** `RP/0/0/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.

	Command or Action	Purpose
Step 7	**neighbor** *ip-address* **pw-id** *pseudowire-id* **Example:** `RP/0/0/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 8	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# 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 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 on page MPC-35 and "Configuring Dynamic Point-to-Point Cross-Connects" section on page MPC-37) 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, 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.subinterface* **l2transport**
3. **service-policy** *[***input** | **output***] [policy-map-name]*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters the configuration mode.

Step 2	**`interface`** *`type interface-path-id.subinterface`* **`l2transport`**
	Example:
	`RP/0/0/CPU0:router(config)# interface GigabitEthernet0/0/0/0.1`
		Configures an interface or connection for L2 switching and specifies the interface attachment circuit.

	Command or Action	Purpose
Step 3	**service-policy** [**input** | **output**] [*policy-map-name*] **Example:** `RP/0/0/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	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 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; and the l2transport command must follow the interface type on the same CLI line (for example, “interface GigabitEthernet0/0/0/0.1 l2transport”).

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id.subinterface* **l2transport**
3. **service-policy** [**input** | **output**] [*policy-map-name*]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters the configuration mode.

Step 2	**`interface`** *`type interface-path-id.subinterface`* **`l2transport`**
	
	Example:
	`RP/0/0/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/0/0/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	**`end`**
	or
	`commit`
	
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 an L2VPN Quality of Service Policy in Frame Relay Mode

This procedure describes how to configure a L2VPN QoS policy in Frame Relay mode.

SUMMARY STEPS

1. **configure**
2. **class-map match-any [***new class map name]*

3. **match frame-relay dlci** [*dlci number*]
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters the configuration mode.

Step 2	**`class-map match any`** *`new class name`*
	Example:
	`RP/0/0/CPU0:router(config-cmap)# class-map match-any A`
		Matches the class map type to a new class map.

Step 3	**`match frame-relay dlci`** *`dlci number`*
	Example:
	`RP/0/0/CPU0:router(config-cmap)# match frame-relay dlci 100-200 500`
		Applies the quality of service on the main interface with a frame relay encapsulation type.

Step 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-cmap)# end`
	or
	`RP/0/0/CPU0:router(config-cmap)# 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 Preferred Tunnel Path

This procedure describes how to configure a preferred tunnel path.

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.

Step 3	**pw-class** {*name*}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# pw-class path1`
		Configures the pseudowire class name.

Step 4	**encapsulation mpls** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls`	Configures the pseudowire encapsulation to MPLS.

	Command or Action	Purpose
Step 5	**preferred-path** {**interface**} {**tunnel-ip** *value* | **tunnel-te** *value* | ***tunnel-tp*** *value*} [**fallback disable**] **Example:** `RP/0/0/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 tunnel is configured as the preferred path goes down, the corresponding pseudowire can also go down.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-mpls)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-mpls-if)# 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 AToM IP Interworking

To configure AToM IP interworking, you need to configure attachment circuits (AC), pseudowire class, and cross connects.

- Configuring Ethernet ACs for AToM IP Interworking, page VPC-46
- Configuring Frame Relay ACs for AToM IP Interworking, page VPC-47
- Configuring ATM AAL5 ACs for AToM IP Interworking, page VPC-49
- Configuring PPP ACs for AToM IP Interworking, page VPC-51
- Configuring Local Switching on PPP ACs, page VPC-52
- Configuring IP Interworking on PPP ACs, page VPC-54
- Configuring cHDLC ACs for AToM IP Interworking, page VPC-56
- Configuring Local Switching on cHDLC ACs, page VPC-57
- Configuring IP Interworking on cHDLC ACs, page VPC-59
- Configuring Frame Relay AC for Bridged Interworking, page VPC-61

Configuring Ethernet ACs for AToM IP Interworking

Perform this task to configure an Ethernet AC for AToM IP Interworking.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **l2transport**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`interface type interface-path-id`
	Example:
	`RP/0/0/CPU0:router(config)# interface ethernet 0/0/0/0`
		Configures the Ethernet interface.

	Command or Action	Purpose
Step 3	**l2transport** **Example:** `RP/0/0/CPU0:router(config-if)# l2transport`	Configures the Layer 2 Transport type for the AC.

Step 4	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 Frame Relay ACs for AToM IP Interworking

Perform this task to configure a Frame Relay AC for AToM IP Interworking.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **encapsulation frame-relay** *frame-relay networks*
4. **frame-relay [intf-type] dce**
5. **interface** *type interface-path-id* **l2transport**
6. **pvc** *number*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface POS 0/2/0/1`	Configures the Layer 2 transport sub-interface.
Step 3	**encapsulation frame-relay** *frame-relay networks* **Example:** `RP/0/0/CPU0:router(config-if)# encapsulation frame-relay`	Encapsulates the Frame Relay network using RFC1490 or RFC2427 encapsulation.
Step 4	**frame-relay [intf-type] dce** **Example:** `RP/0/0/CPU0:router(config-if)# frame`	Configures Frame Relay interface type based on the DCE mode.
Step 5	**interface** *type interface-path-id* **l2transport** **Example:** `RP/0/0/CPU0:router(config)# interface POS 0/2/0/1.200 l2transport`	Configures the Layer 2 transport sub-interface.

	Command or Action	Purpose
Step 6	**pvc** *number* **Example:** `RP/0/0/CPU0:router(config-subif)# pvc 20`	Configures a virtual circuit.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 ATM AAL5 ACs for AToM IP Interworking

Perform this task to configure an ATM AAL5 AC for AToM IP Interworking.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id* **l2transport**
3. **pvc** *number*
4. **encapsulation {aal5mux} {ipv4}**
5. **Repeat steps 1 through 3**
6. **encapsulation {aal5snap}**
7. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **l2transport** **Example:** `RP/0/0/CPU0:router(config)# interface ATM 0/2/0/1.200 l2transport`	Configures the Layer 2 transport sub-interface.
Step 3	**pvc** *number* **Example:** `RP/0/0/CPU0:router(config-subif)# pvc 2/200`	Configures a virtual circuit.

Step 4	**encapsulation {aal5mux} {ipv4}**
	Example:
	`RP/0/0/CPU0:router(config-atm-l2transport-pvc)# encapsulation aal5mux ipv4`
		Configures the AAL5 MUX ATM encapsulation over an IPv4 network.

Step 5	**encapsulation {aal5snap}**
	Example:
	`RP/0/0/CPU0:router(config-atm-l2transport-pvc)# encapsulation aal5snap`
		Configures the AAL5 SNAP ATM encapsulation.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-atm-l2transport-pvc)# end`
	or
	`RP/0/0/CPU0:router(config-atm-l2transport-pvc)# 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 PPP ACs for AToM IP Interworking

Perform this task to configure a PPP AC for AToM IP Interworking.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **encapsulation ppp**
4. **ppp ipcp proxy-address** *ip_address*
5. **l2transport**
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface POS 0/2/0/1`	Configures the Layer 2 transport interface.
Step 3	**encapsulation ppp** **Example:** `RP/0/0/CPU0:router(config-if)# encapsulation ppp`	Enables PPP encapsulation.
Step 4	**ppp ipcp proxy-address** *ip_address* **Example:** `RP/0/0/CPU0:router(config-if)# ppp ipcp proxy-address 1.2.3.4`	Configures IP address of the remote CE router. This IP address is used by the PE router during IPCP negotiations with the CE router.

	Command or Action	Purpose
Step 5	**l2transport** **Example:** `RP/0/0/CPU0:router(config-if)# l2transport`	Configures Layer 2 transport.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 Switching on PPP ACs

Perform this task to configure local switching on PPP ACs.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **interworking ipv4**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.
Step 3	**xconnect group**-*name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group group1`	Specifies the name of the cross-connect group.
Step 4	**p2p** *xconnect-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p bar`	Specifies a name for the point-to-point cross-connect.
Step 5	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface POS 0/2/0/1`	Specifies the interface type ID.

	Command or Action	Purpose

Step 6	**`interworking ipv4`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# interworking ipv4`
		Specifies the interface type ID.

Step 7	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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 IP Interworking on PPP ACs

Perform this task to configure IP Interworking on PPP ACs.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **neighbor** *ip-address* **pw-id** *pseudowire-id*
7. **interworking ipv4**
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.
Step 3	**xconnect group** *group-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group group1`	Specifies the name of the cross-connect group.
Step 4	**p2p** *xconnect-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p bar`	Specifies a name for the point-to-point cross-connect.
Step 5	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface POS 0/2/0/1`	Specifies the interface type ID.
Step 6	**neighbor** *ip-address* **pw-id** *pseudowire-id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000`	Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose

Step 7	**`interworking ipv4`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# interworking ipv4`
		Specifies the interface type ID.

Step 8	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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 cHDLC ACs for AToM IP Interworking

Perform this task to configure a cHDLC AC for AToM IP Interworking.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **l2transport**
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface POS 0/2/0/1`
		Configures the Layer 2 transport interface.

Step 3	**`l2transport`**
	Example:
	`RP/0/0/CPU0:router(config-if)# l2transport`
		Configures Layer 2 transport.

Step 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 Switching on cHDLC ACs

Perform this task to configure local switching on cHDLC ACs.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*

5. **interface** *type interface-path-id*
6. **interworking ipv4**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enters L2VPN configuration mode.

Step 3	**`xconnect group`** *`group-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group group1`
		Specifies the name of the cross-connect group.

Step 4	**`p2p`** *`xconnect-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p bar`
		Specifies a name for the point-to-point cross-connect.

Step 5	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface POS 0/2/0/1`
		Specifies the interface type ID.

	Command or Action	Purpose

Step 6	**`interworking ipv4`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# interworking ipv4`
		Specifies the interface type ID.

Step 7	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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 IP Interworking on cHDLC ACs

Perform this task to configure IP Interworking on cHDLC ACs.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface** *type interface-path-id*
6. **neighbor** *ip-address* **pw-id** *pseudowire-id*
7. **interworking ipv4**
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enters L2VPN configuration mode.

Step 3	**`xconnect group`** *`group-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group group1`
		Specifies the name of the cross-connect group.

Step 4	**`p2p`** *`xconnect-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p bar`
		Specifies a name for the point-to-point cross-connect.

Step 5	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface POS 0/2/0/1`
		Specifies the interface type ID.

Step 6	**`neighbor`** *`ip-address`* **`pw-id`** *`pseudowire-id`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000`
		Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose
Step 7	**interworking ipv4** **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# interworking ipv4`	Specifies the type of interworking (routed or bridged interworking).

Step 8	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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 Frame Relay AC for Bridged Interworking

Perform this task to configure a Frame Relay AC for Bridged Interworking.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **encapsulation frame-relay** *frame-relay networks*
4. **load-interval** *interval*
5. **frame-relay intf-type**
6. **frame-relay lmi disable**
7. **interface type instance-path-id l2transport**
8. **pvc** *number*
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface POS 0/2/0/1`
		Configures the Layer 2 transport interface.

Step 3	**`encapsulation frame-relay`** *`frame-relay networks`*
	Example:
	`RP/0/0/CPU0:router(config-if)# encapsulation frame-relay`
		Encapsulates the Frame Relay network using RFC1490 or RFC2427 encapsulation.

Step 4	**`load-interval`** *`interval`*
	Example:
	`RP/0/0/CPU0:router(config-if)# load interval 30`
		Sets the length of time for which data is used for load calculations.

Step 5	**`frame-relay intf-type {dce | dte}`**
	Example:
	`RP/0/0/CPU0:router(config-if)# frame-relay intf-type`
		Configures the type of support provided by the interface.
		• If your router functions as a switch connected to another router, use the frame-relay intf-type dce command to configure the LMI type to support data communication equipment (DCE).
		• If your router is connected to a Frame Relay network, use the **frame-relay intf-type dte** command to configure the LMI type to support data terminal equipment (DTE).
		Note The default interface type is DTE.

Step 6	**`frame-relay lmi disable`**
	Example:
	`RP/0/0/CPU0:router(config-if)# frame-relay lmi disable`
		Disables local management interface (LMI).

Step 7	**`interface`** *`type interface-path-id`* **`l2transport`**
	Example:
	`RP/0/0/CPU0:router(config-if)# interface POS 0/2/0/1.200 l2transport`
		Configures the Layer 2 transport sub-interface

	Command or Action	Purpose
Step 8	**pvc** *number* **Example:2** `RP/0/0/CPU0:router(config-if)# pvc 20`	Configures a virtual circuit.

Step 9	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-fr-vc)# end`
	or
	`RP/0/0/CPU0:router(config-fr-vc)# 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 Pseudowire Class

Perform this task to configure a pseudowire class.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation mpls**
5. **protocol ldp**
6. **vccv**
7. **end**
   or
   commit

DETAILED STEPS

	Command	Purpose
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 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	**vccv** **Example:** `RP/0/RSP0/CPU0:router (config-l2vpn-pwc-encap-mpls)# vccv ver none`	Configures virtual circuit connection verification (VCCV) settings.
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.

Configuring Circuit Emulation Over Packet Switched Network

Perform these tasks to configure CEoP:

- Adding CEM attachment circuit to a Pseudowire, page VPC-65
- Associating a Pseudowire Class, page VPC-66
- Configuring a Backup Pseudowire, page VPC-69

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 ip-address* **pw-id** *pseudowire-id*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enters L2VPN configuration mode.

Step 3	**`xconnect group`** *`group-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp_1`
		Enters the name of the cross-connect group.

Step 4	**`p2p`** *`xconnect-name`*
	Example:
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface CEM0/1/0/9:10`
		Specifies the interface type and instance.

	Command or Action	Purpose
Step 6	`neighbor` *A.B.C.D* **pw-id** *pseudowire-id* **Example:** `RP/0/0/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.

Step 7	`end`
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# 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.

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 ip-address* **pw-id** *pseudowire-id*
11. **pw-class** *class-name*

12. **end**
or
commit

DETAILED STEPS

	Command	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router (config)# l2vpn`
		Enters Layer 2 VPN configuration mode.

Step 3	**`pw-class`** *`class-name`*
	Example:
	`RP/0/0/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/0/0/CPU0:router (config-l2vpn-pwc)# encapsulation mpls`
		Sets pseudowire encapsulation to MPLS.

Step 5	**`protocol ldp`**
	Example:
	`RP/0/0/CPU0:router (config-l2vpn-pwc-encap-mpls)# protocol ldp`
		Sets pseudowire signaling protocol to LDP.

Step 6	**`end`**
	Example:
	`RP/0/0/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.

	Command	Purpose

Step 7	**`xconnect group`** *`group-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp_1`
		Configures a cross-connect group.

Step 8	**`p2p`** *`xconnect-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p vlan1`
		Configures a point-to-point cross-connect.

Step 9	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/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/0/0/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.

	Command	Purpose
Step 11	**pw-class** *class-name* **Example:** `RP/0/0/CPU0:router (config-l2vpn-xc-p2p)# pw-class class_cem`	Associates the P2P attachment circuit with the specified pseudowire class.

Step 12	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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 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** {*A.B.C.D*} {**pw-id** *value*}
6. **backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/0/CPU0:router# configure	Enters configuration mode.
Step 2	**l2vpn** **Example:** RP/0/0/CPU0:router(config)# l2vpn RP/0/0/CPU0:router(config-l2vpn)#	Enters L2VPN configuration mode.
Step 3	**xconnect group** *group-name* **Example:** RP/O/0/CPU0:router(config-l2vpn)# xconnect group A RP/0/0/CPU0:router(config-l2vpn-xc)#	Enters the name of the cross-connect group.
Step 4	**p2p** {*xconnect-name*} **Example:** RP/0/0/CPU0:router(config-l2vpn-xc)# p2p xc1 RP/0/0/CPU0:router(config-l2vpn-xc-p2p)#	Enters a name for the point-to-point cross-connect.
Step 5	**neighbor** {*A.B.C.D*} {**pw-id** *value*} **Example:** RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.2 pw-id 11	Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose

Step 6	**backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 10.2.2.2 pw-id 5`
	`RP/0/0/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 cross-connect. The IP address argument (*A.B.C.D*) 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 7	**end**
	or
	commit
	
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# end`
	or
	`RP/0/0/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.

Configuring L2VPN Nonstop Routing

Perform this task to configure L2VPN Nonstop Routing.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **nsr**
4. **logging nsr**
5. **end**
or
commit

DETAILED STEPS

	Command	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router (config)# l2vpn`
		Enters Layer 2 VPN configuration mode.

Step 3	**`nsr`**
	Example:
	`RP/0/0/CPU0:router (config-l2vpn)# nsr`
		Enables L2VPN nonstop routing.

	Command	Purpose
Step 4	**logging nsr** **Example:** `RP/0/0/CPU0:router (config-l2vpn)# logging nsr`	Enables logging of NSR events.

Step 5	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# end`
	or
	`RP/0/0/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 Pseudowire Grouping

Perform this task to enable pseudowire grouping.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose

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.

	Command or Action	Purpose
Step 3	**pw-grouping** **Example:** `RP/0/RSP0/CPU0:router(config-l2vpn)# pw-grouping`	Enables pseudowire grouping

Step 4	**end**
	or
	commit
	Example:
	`RP/0/RSP0/CPU0:router(config-l2vpn)# end`
	or
	`RP/0/RSP0/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.

Configuration Examples for L2VPN

In this example, two traffic classes are created and their match criteria are defined. For the first traffic class called class1, ACL 101 is used as the match criterion. For the second traffic class called class2, ACL 102 is used as the match criterion. Packets are checked against the contents of these ACLs to determine if they belong to the class.

This section includes these configuration examples:

- L2VPN Interface Configuration: Example, page VPC-76
- Point-to-Point Cross-connect Configuration: Examples, page VPC-76
- Inter-AS: Example, page VPC-77
- L2VPN Quality of Service: Example, page VPC-79
- Preferred Path: Example, page VPC-79
- AToM IP Interworking: Examples, page VPC-79
- AToM Cross Connect Configuration: Example, page VPC-82
- Configuring L2VPN over GRE Tunnels: Example, page VPC-82

L2VPN Interface Configuration: Example

The following example shows how to configure an L2VPN interface:

```
configure
 interface GigabitEthernet0/0/0/0.1 l2transport
 dot1q vlan 1
 end
```

Point-to-Point Cross-connect Configuration: Examples

This section includes configuration examples for both static and dynamic point-to-point cross-connects.

Static Configuration

The following example shows how to configure a static point-to-point cross-connect:

```
configure
 l2vpn
  xconnect group vlan_grp_1
   p2p vlan1
   interworking ipv4
   interface GigabitEthernet0/0/0/0.1
   neighbor 2.2.2.2 pw-id 2000
    mpls static label local 699 remote 890
    commit
```

Dynamic Configuration

The following example shows how to configure a dynamic point-to-point cross-connect:

```
configure
 l2vpn
  xconnect group vlan_grp_1
   p2p vlan1
   interworking ipv4
   interface GigabitEthernet0/0/0/0.1
   neighbor 2.2.1.1 pw-id 1commit
```

The following example shows how to configure a dynamic point-to-point cross-connect using OSPF and MPLS LDP:

```
configure
l2vpn
 pw-class ceop
  encapsulation mpls
 !
 xconnect group SATOP
  p2p STP1
   interface CEM0/2/1/0/1/1/1/1
   neighbor 24.24.24.2 pw-id 1001
    pw-class ceop
   !
 xconnect group CESOPSN
  p2p CSPN1
   interface CEM0/2/1/0/1/1/1/2:0
   neighbor 24.24.24.2 pw-id 1002
    pw-class ceop
   !


show runn router ospf
router ospf 10
 router-id 21.21.21.1
 area 0
  interface Loopback0
  !
  interface GigabitEthernet0/2/2/0 <<< Core Facing Interface
  !
 !
!
RP/0/RSP0/CPU0:CEOP-03#
RP/0/RSP0/CPU0:CEOP-03#
RP/0/RSP0/CPU0:CEOP-03#show runn mpls ldp mpls ldp
 graceful-restart                 <<<< required to avoid drops during L2VPN_MGR process
restarts
 interface GigabitEthernet0/2/2/0  <<< Core Facing Interface  !
!
```

Inter-AS: Example

The following example shows how to set up an AC to AC cross-connect from AC1 to AC2:

```
router-id Loopback0

interface Loopback0
 ipv4 address 127.0.0.1 255.255.255.0
!
interface GigabitEthernet0/1/0/0.1 l2transport dot1q vlan 1!
!
interface GigabitEthernet0/0/0/3
 ipv4 address 127.0.0.1 255.255.255.0
 keepalive disable
!
interface GigabitEthernet0/0/0/4
 ipv4 address 127.0.0.1 255.255.255.0
 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 40.0.0.5
  remote-as 100
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family ipv4 labeled-unicast
  !
 !
!
l2vpn
 xconnect group xc1
  p2p ac2ac1
   interface GigabitEthernet0/1/0/0.1
   neighbor 20.0.0.5 pw-id 101
  !
  p2p ac2ac2
   interface GigabitEthernet0/1/0/0.2
   neighbor 20.0.0.5 pw-id 102
  !
  p2p ac2ac3
   interface GigabitEthernet0/1/0/0.3
   neighbor 20.0.0.5 pw-id 103
  !
  p2p ac2ac4
   interface GigabitEthernet0/1/0/0.4
   neighbor 20.0.0.5 pw-id 104
  !
  p2p ac2ac5
   interface GigabitEthernet0/1/0/0.5
   neighbor 20.0.0.5 pw-id 105
  !
  p2p ac2ac6
   interface GigabitEthernet0/1/0/0.6
   neighbor 20.0.0.5 pw-id 106
  !
  p2p ac2ac7
   interface GigabitEthernet0/1/0/0.7
   neighbor 20.0.0.5 pw-id 107
  !
  p2p ac2ac8
   interface GigabitEthernet0/1/0/0.8
   neighbor 20.0.0.5 pw-id 108
  !
  p2p ac2ac9
   interface GigabitEthernet0/1/0/0.9
   neighbor 20.0.0.5 pw-id 109
  !
  p2p ac2ac10
   interface GigabitEthernet0/1/0/0.10
   neighbor 20.0.0.5 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

The following 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 | output] [policy-map-name]
commit
```

Preferred Path: Example

The following example shows how to configure preferred tunnel path:

```
configure
 l2vpn
 pw-class path1
  encapsulation mpls
   preferred-path interface tunnel-ip value fallback disable
```

AToM IP Interworking: Examples

This section includes configuration examples for all supported AC modes in AToM IP Interworking.

Ethernet

```
interface GigabitEthernet0/0/0/2
  l2transport
!
interface GigabitEthernet0/0/0/3.1 l2transport
  dot1q vlan 1
!
interface GigabitEthernet0/0/0/3.2 l2transport
dot1q vlan 2 2
```

Frame Relay

```
interface POS0/2/0/1
mtu 1500
encapsulation frame-relay
frame-relay intf-type dce
!
interface POS0/2/0/1.20 l2transport
pvc 20
!
```

ATM AAL5

```
interface ATM0/3/0/1.200 l2transport
pvc 20/200
encapsulation aal5mux ipv4
!
interface ATM0/3/0/1.300 l2transport
```

```
pvc 30/300
encapsulation aal5snap
!
interface ATM0/3/0/1.300 l2transport
pvc 30/400
encapsulation aal5nlpid
```

PPP

```
interface POS0/0/0/0
 encapsulation ppp
 ppp ipcp proxy-address 1.2.3.4
 l2transport
 !
!
interface POS0/0/0/1
 ppp ipcp proxy-address 1.2.3.14
 encapsulation ppp
 l2transport
 !
!

l2vpn
 xconnect group foo
  p2p bar
   interface POS0/0/0/0
   interface POS0/0/0/1
   interworking ipv4
  !
 !
!

l2vpn
 xconnect group foo
  p2p bar
   interface POS0/0/0/0
   neighbor 10.1.1.1 pw-id 666
   interworking ipv4
```

cHDLC

```
interface pos 0/1/0/1
l2transport

interface pos 0/1/0/2
l2transport


l2vpn
 xconnect group foo
  p2p bar
   interface POS 0/1/0/1
   interface POS 0/1/0/2
   interworking ipv4
  !
 !
!
l2vpn
 xconnect group foo
  p2p bar
```

```
     interface POS 0/1/0/1
     neighbor 10.1.1.1 pw-id 666
     interworking ipv4
    !
   !
```

Bridged Interworking: Example

```
interface POS0/2/0/1
 mtu 1504
 encapsulation frame-relay
 load-interval 30
 frame-relay intf-type dce
 frame-relay lmi disable
!
interface POS0/2/0/1.20 l2transport
 pvc 20
```

ATM AAL5 to Ethernet Bridged Interworking: Example

ATM side:

```
controller T3 0/4/3/1
mode atm
!
interface ATM0/4/3/1.1 l2transport
 pvc 50/50
  encapsulation aal5snap
 !
 mtu 1500
!
l2vpn
pw-class mpls_class
 encapsulation mpls
  protocol ldp
 !
!
xconnect group pe1_to_pe2
 p2p xc2
  interface ATM0/4/3/1.1
  neighbor 5.5.5.5 pw-id 2
   pw-class mpls_class
  !
  interworking ethernet
 !
!
```

Ethernet side:

```
l2vpn
pw-class mpls_class
 encapsulation mpls
  protocol ldp
 !
!
```

```
interface GigabitEthernet0/0/0/0.1 l2transport  dot1q vlan 1  end !
xconnect group pe1_to_pe2
 p2p xc2
  interface GigabitEthernet0/3/0/0.1
  neighbor 2.2.2.2 pw-id 2
   pw-class mpls_class
  !
  interworking ethernet
 !
!
```

AToM Cross Connect Configuration: Example

This section includes configuration examples for all supported AToM Cross Connects.

```
l2vpn
pseudowire-class ipiw
  encapsulation mpls
!
xconnect group port
  p2p port1
    interface GigabitEthernet0/0/0/2
    neighbor 11.11.11.11 pw-id 300 pw-class ipiw
  !
!
xconnect group vlan
  p2p vlan1
    interface GigabitEthernet0/0/0/3.1
    neighbor 11.11.11.11 pw-id 400 pw-class ipiw
  !
!
xconnect group frame-relay
  p2p frame1
    interface POS0/2/0/1.20
    neighbor 11.11.11.11 pw-id 600 pw-class ipiw
  !
!
xconnect group atm
  p2p atm1
    interface ATM0/3/0/1.200
    neighbor 11.11.11.11 pw-id 700 pw-class ipiw
  !
  p2p atm2
    interface ATM0/3/0/1.300
    neighbor 11.11.11.11 pw-id 800 pw-class ipiw
```

Configuring L2VPN over GRE Tunnels: Example

The following example shows how to configure L2VPN over GRE tunnels:

```
interface tunnel-ip101
 ipv4 address 150.10.1.204 255.255.255.0
 ipv6 address 150:10:1::204/64
 tunnel mode gre ipv4
 tunnel source Loopback1
 tunnel destination 100.1.1.202

router ospf 1
```

```
 router-id 100.0.1.204
 cost 1
 router-id Loopback0
 area 1
  interface Loopback0
  !
  interface tunnel-ip101

mpls ldp
 router-id 100.0.1.204
 interface tunnel-ip101

l2vpn
 xconnect group pe2
  p2p 2001
   interface GigabitEthernet0/2/0/0.2001
   neighbor 100.0.1.202 pw-id 2001
```

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
```

Additional References

For additional information related to implementing MPLS Layer 2 VPN, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS VPN-related commands	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*

Related Topic	Document Title
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS Layer 3 VPNs	*Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
Cisco CRS router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	—

[1] Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 4447	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*, April 2006
RFC 4448	*Encapsulation Methods for Transport of Ethernet over MPLS Networks*, April 2006

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing Virtual Private LAN Services

This module provides the conceptual and configuration information for Virtual Private LAN Services (VPLS) on Cisco IOS XR software. 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, Layer 2 VPNs.

For MPLS Layer 2 virtual private networks (VPNs), see Implementing MPLS Layer 2 VPNs module.

Note For more information about MPLS Layer 2 VPN on Cisco IOS XR software and for descriptions of the commands listed in this module, see the "Related Documents" section. To locate documentation for other commands that might appear while executing a configuration task, search online in the Cisco IOS XR software master command index.

Feature History for Implementing Virtual Private LAN Services on Cisco IOS XR Configuration Module

Release	Modification
Release 3.7.0	This feature was introduced.
Release 3.8.0	Support for the bridging funtionality feature (VPLS based) and pseudowire redundancy, was added on the Cisco CRS-1 router.
Release 3.9.0	The following features were added: • Blocking unknown unicast flooding. • Disabling MAC flush.
Release 4.1.1	Support for these features was added: • Mutlisegment Pseudowire • Pseudowire Redundancy • Pseudowire Headend
Release 4.2.1	The pseudowire headend (PWHE) feature was enhanced to support: • eBGP on PWHE interfaces • IPv6 Unicast on PWHE
Release 4.3.0	Support was added for these features: • Flow Aware Transport (FAT) Pseudowire • Pseudowire Grouping

Contents

Prerequisites for Implementing Virtual Private LAN Services

Before you configure VPLS, ensure that the network is configured as follows:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

  If you need assistance with your task group assignment, contact your system administrator.
- Configure IP routing in the core so that the provider edge (PE) routers can reach each other through IP.
- Configure MPLS and Label Distribution Protocol (LDP) in the core so that a label switched path (LSP) exists between the PE routers.
- Configure a loopback interface to originate and terminate Layer 2 traffic. Make sure that the PE routers can access the other router's loopback interface.

Note The loopback interface is not needed in all cases. For example, tunnel selection does not need a loopback interface when VPLS is directly mapped to a TE tunnel.

Restrictions for Implementing Virtual Private LAN Services

The following restrictions are listed for implementing VPLS:

- All attachment circuits in a bridge domain on an Engine 3 line card must be the same type (for example, port, dot1q, qinq, or qinany), value (VLAN ID), and EtherType (for example, 0x8100, 0x9100, or 0x9200).
- The Engine 3 line cards, cannot simultaneously have attachment circuits and MPLS-enabled on any one of its interfaces. The line card cannot be Edge-facing and Core-facing at the same time.
- The line card requires ternary content addressable memory (TCAM) Carving configuration.
- Virtual Forwarding Instance (VFI) names have to be unique, because a bridge domain can have only one VFI.
- A PW cannot belong to both a peer-to-peer (P2P) cross-connect group and a VPLS bridge-domain. This means that the neighboring IP address and the pseudowire ID have to be unique on the router, because the pseudowire ID is signaled to the remote provider edge.

- You cannot manually set up a PW on one PE and use auto-discovery on the other PE to configure the same PW in the other direction.

For the Engine 5 line card, version 1 of the Ethernet SPA does not support QinQ mode and QinAny mode.

Note For the Engine 5 line card, version 2 of the Ethernet SPA supports all VLAN modes, such as VLAN mode, QinQ mode, or QinAny mode.

Information About Implementing Virtual Private LAN Services

To implement Virtual Private LAN Services (VPLS), you should understand the following concepts:

Virtual Private LAN Services Overview

Virtual Private LAN Service (VPLS) enables geographically separated local-area network (LAN) segments to be interconnected as a single bridged domain over an MPLS network. The full functions of the traditional LAN such as MAC address learning, aging, and switching are emulated across all the remotely connected LAN segments that are part of a single bridged domain. A service provider can offer VPLS service to multiple customers over the MPLS network by defining different bridged domains for different customers. Packets from one bridged domain are never carried over or delivered to another bridged domain, thus ensuring the privacy of the LAN service.

VPLS transports Ethernet 802.3, VLAN 802.1q, and VLAN-in-VLAN (Q-in-Q) traffic across multiple sites that belong to the same Layer 2 broadcast domain. VPLS offers simple Virtual LAN 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 provider edge (PE) routers. The VPLS implementation is based on Label Distribution Protocol (LDP)-based pseudowire signaling.

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

After provisioning attachment circuits, neighbor relationships across the MPLS network for this specific instance are established through a set of manual commands identifying the end PEs. When the neighbor association is complete, a full mesh of pseudowires is established among the network-facing provider edge devices, which is a gateway between the MPLS core and the customer domain.

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

VPLS requires the provider edge device to be MPLS-capable. 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.

Note VPLS with Traffic Engineering Fast Reroute (TE FRR) is not supported.

VPLS for an MPLS-based Provider Core

VPLS is a multipoint Layer 2 VPN technology that connects two or more customer devices using bridging techniques. The VPLS architecture allows for the end-to-end connection between the Provider Edge (PE) routers to provide Multipoint Ethernet Services.

VPLS requires the creation of a bridge domain (Layer 2 broadcast domain) on each of the PE routers. The access connections to the bridge domain on a PE router are called *attachment circuits* (AC).

The attachment circuits can be a set of physical ports, virtual ports, or both that are connected to the bridge at each PE device in the network.

The MPLS/IP provider core simulates a virtual bridge that connects the multiple attachment circuits on each of the PE devices together to form a single broadcast domain. A VFI is created on the PE router for each VPLS instance. The PE routers make packet-forwarding decisions by looking up the VFI of a particular VPLS instance. The VFI acts like a virtual bridge for a given VPLS instance. More than one attachment circuit belonging to a given VPLS are connected to the VFI. The PE router establishes emulated VCs to all the other PE routers in that VPLS instance and attaches these emulated VCs to the VFI. Packet forwarding decisions are based on the data structures maintained in the VFI.

Hierarchical VPLS

Hierarchical VPLS (H-VPLS) is an extension of basic VPLS that provides scaling and operational benefits. H-VPLS provides a solution to deliver Ethernet multipoint services over MPLS. H-VPLS partitions a network into several edge domains that are interconnected using an MPLS core. The use of Ethernet switches at the edge offers significant technical and economic advantages. H-VPLS also allows Ethernet point-to-point and multipoint Layer 2 VPN services, as well as Ethernet access to high-speed Internet and IP VPN services.

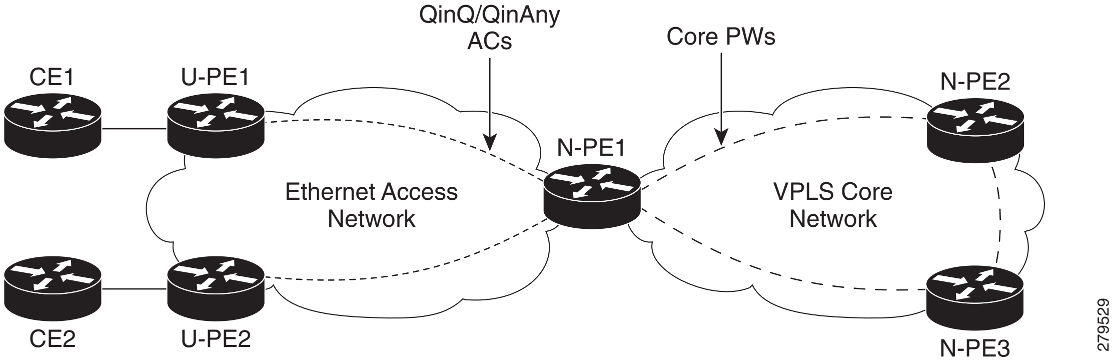
Two flavors of H-VPLS are:

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

H-VPLS with Ethernet Access QinQ or QinAny

Figure 1 shows Ethernet access for H-VPLS. The edge domain can be built using Ethernet switches and techniques such as QinQ. Using Ethernet as the edge technology simplifies the operation of the edge domain and reduces the cost of the edge devices.

Figure 13 Ethernet Access for H-VPLS

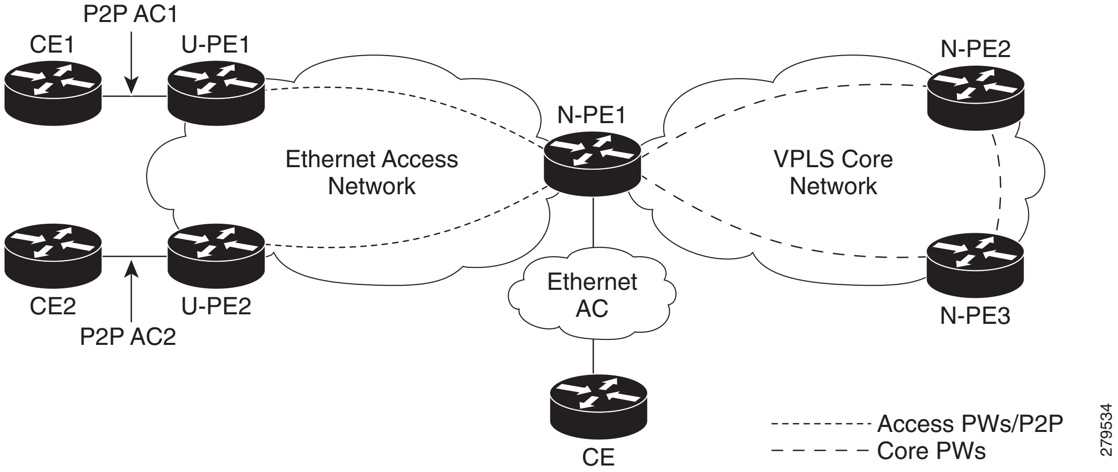


H-VPLS with PW-access

Figure 14 shows pseudowire (PW) access for H-VPLS. The edge domain can be an MPLS access network. In this scenario, the U-PE device carries the customer traffic from attachment circuits (AC) over the point to point (p2p) pseudowires. The p2p pseudowires terminate in a bridge domain configured on the N-PE device.

Access PW is configured as a member directly under a bridge domain. A bridge-domain in N-PE1 can have multiple ACs (physical/VLAN Ethernet ports), multiple access PWs and one VFI (consisting of core PWs) as members, is depicted in Figure 14.

Figure 14 PW access for H-VPLS



Signaling

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, often referred to as *signaling mechanisms*.

The implementation of VPLS in a network requires the establishment of a full mesh of pseudowires between the provider edge (PE) routers. The signaling of pseudowires between provider edge devices, described in *draft-ietf-l2vpn-vpls-ldp-09*, uses targeted LDP sessions to exchange label values and attributes and to setup the pseudowires. LDP is an efficient mechanism for signaling pseudowire status for Ethernet point-to-point and multipoint services.

Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling

The Cisco IOS Software encodes the NLRI length in the fist byte in bits format in the BGP Update message. However, the Cisco IOS XR Software interprets the NLRI length in 2 bytes. Therefore, when the BGP neighbor with VPLS-VPWS address family is configured between the IOS and the IOS XR, NLRI mismatch can happen, leading to flapping between neighbors. To avoid this conflict, IOS supports **prefix-length-size 2** command that needs to be enabled for IOS to work with IOS XR. When the **prefix-length-size 2** command is configured in IOS, the NLRI length is encoded in bytes. This configuration is mandatory for IOS to work with IOS XR.

This is a sample IOS configuration with the **prefix-length-size 2** command:

```
router bgp 1
 address-family l2vpn vpls
  neighbor 5.5.5.2 activate
  neighbor 5.5.5.2 prefix-length-size 2 --------> NLRI length = 2 bytes
 exit-address-family
```

Bridge Domain

The native bridge domain refers to a Layer 2 broadcast domain consisting of a set of physical or virtual ports (including VFI). Data frames are switched within a bridge domain based on the destination MAC address. Multicast, broadcast, and unknown destination unicast frames are flooded within the bridge domain. In addition, the source MAC address learning is performed on all incoming frames on a bridge domain. A learned address is aged out. Incoming frames are mapped to a bridge domain, based on either the ingress port or a combination of both an ingress port and a MAC header field.

By default, split horizon is enabled on a bridge domain. In other words, any packets that are coming on either the attachment circuits or pseudowires are not returned on the same attachment circuits or pseudowires. In addition, the packets that are received on one pseudowire are not replicated on other pseudowires in the same VFI.

MAC Address-related Parameters

The MAC address table contains a list of the known MAC addresses and their forwarding information. In the current VPLS design, the MAC address table and its management are distributed. In other words, a copy of the MAC address table is maintained on the route processor (RP) card and the line cards.

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

- MAC Address Flooding, page VPC-93

MAC Address Flooding

Ethernet services require that frames that are sent to broadcast addresses and to unknown destination addresses be flooded to all ports. To obtain flooding within VPLS broadcast models, all unknown unicast, broadcast, and multicast frames are flooded over the corresponding pseudowires and to all attachment circuits. Therefore, a PE must replicate packets across both attachment circuits and pseudowires.

MAC Address-based Forwarding

To forward a frame, a PE must associate a destination MAC address with a pseudowire or attachment circuit. This type of association is provided through a static configuration on each PE or through dynamic learning, which is flooded to all bridge ports.

Note In this case, split horizon forwarding applies; for example, frames that are coming in on an attachment circuit or pseudowire are not sent out of the same attachment circuit or pseudowire. The pseudowire frames, which are received on one pseudowire, are replicated on to other attachment circuits, VFI pseudowires and access pseudowires.

MAC Address Source-based Learning

When a frame arrives on a bridge port (for example, pseudowire or attachment circuit) and the source MAC address is unknown to the receiving PE router, the source MAC address is associated with the pseudowire or attachment circuit. Outbound frames to the MAC address are forwarded to the appropriate pseudowire or attachment circuit.

MAC address source-based learning uses the MAC address information that is learned in the hardware forwarding path. The updated MAC tables are sent to all line cards (LCs) and program the hardware for the router.

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 limit is set at the bridge domain level and the port level. When the MAC address limit is violated, the system is configured to take one of the actions that are listed in Table 3.

Table 3 MAC Address Limit Actions

Action	Description
Limit flood	Discards the new MAC addresses.
Limit no-flood	Discards the new MAC addresses. Flooding of unknown unicast packets is disabled.
Shutdown	Disables the bridge domain or bridge port. When the bridge domain is down, none of the bridging functions, such as learning, flooding, forwarding, and so forth take place for the bridge domain. If a bridge port is down as a result of the action, the interface or pseudowire representing the bridge port remains up but the bridge port is not participating in the bridge. When disabled, the port or bridge domain is manually brought up by using an EXEC CLI.

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

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

To clear the MAC limit condition, the number of MACs must go below 75 percent of the configured limit.

MAC Address Withdrawal

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

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

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

The LDP MAC Withdrawal feature is generated due to the following 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 and are not propagated over access pseudowires. RFC 4762 specifies that both wildcards (by means of an empty Type, Length and Value [TLV]) and a specific MAC address withdrawal. Cisco IOS XR software supports only a wildcard MAC address withdrawal.

LSP Ping over VPWS and VPLS

For Cisco IOS XR software, the existing support for the Label Switched Path (LSP) ping and traceroute verification mechanisms for point-to-point pseudowires (signaled using LDP FEC128) is extended to cover the pseudowires that are associated with the VFI (VPLS). Currently, the support for the LSP ping and traceroute is limited to manually configured VPLS and access pseudowires (signaled using LDP FEC128). Virtual Circuit Connection Verification (VCCV) is also supported on access pseudowires. For information about VCCV support and the **ping mpls pseudowire** command, see *Cisco IOS XR MPLS Command Reference for the Cisco XR 12000 Series Router*.

Pseudowire Redundancy for P2P AToM Cross-Connects

Backup pseudowires (PW) are associated with the corresponding primary pseudowires. A backup PW is not programmed to forward data when inactive. It is activated only if a primary PW fails. This is known as *pseudowire redundancy*. The primary reason for backing up a PW is to reduce traffic loss when a primary PW fails. When the primary PW is active again, it resumes its activity.

A primary PW can be associated with only one backup PW. Similarly, a backup PW can be associated with only one primary PW.

Note This feature is supported only for an AToM instance on the Cisco XR 12000 Series Router, and for an EoMPLS instance on the Cisco CRS-1 router.

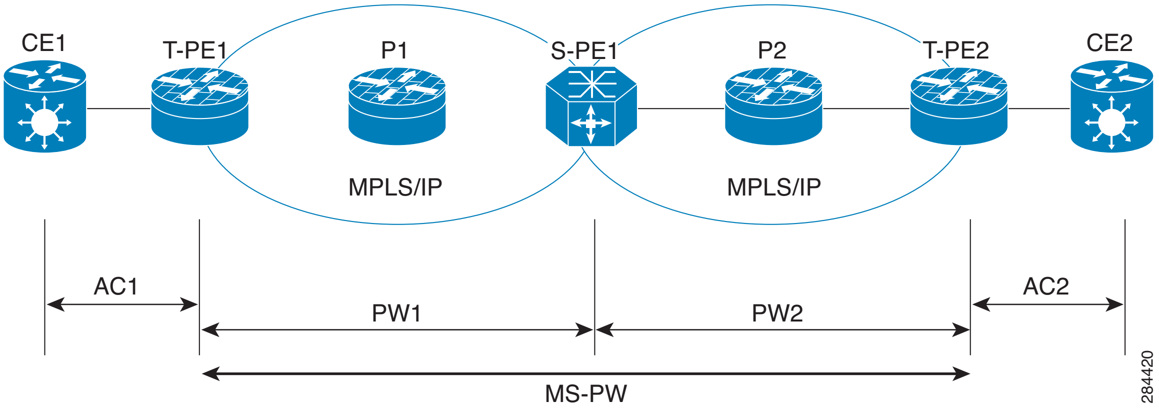
Multisegment Pseudowire

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 15 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.

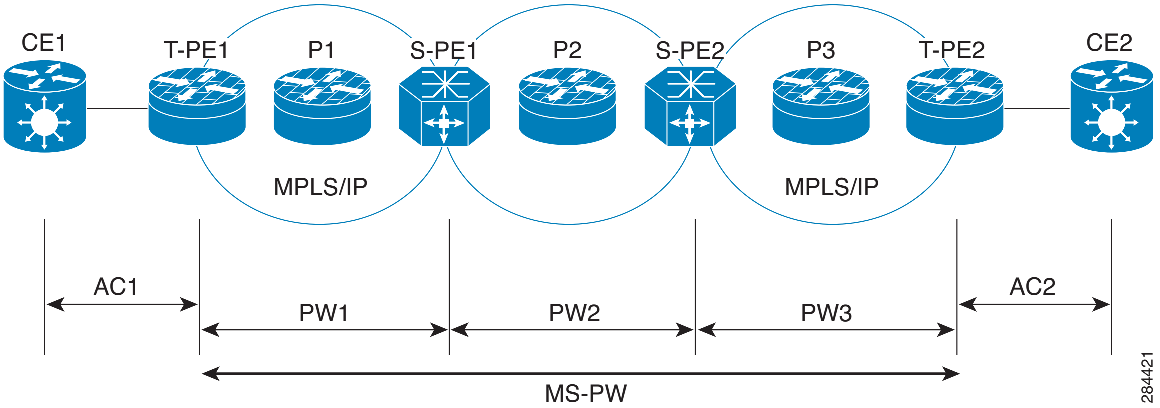
  The PE nodes are connected to networks employing different PW signaling and encapsulation protocols. Sometimes, it is not possible to use a single segment PW.
  A multisegment pseudowire, with the appropriate interworking performed at the PW switching points, enables PW connectivity between the PE nodes in the network.

Pseudowire Switching

Pseudowire Switching connects two or more contiguous pseudowire segments to form an end-to-end multihop pseudowire. This end-to-end pseudowire functions as a single point-to-point pseudowire. It allows you to extend pseudowires across an inter-AS boundary or across two separate networks.

The edge to edge PW may traverse several switching points, in separate administrative domains. For management and troubleshooting reasons you can record information about the switching points that the PW traverses. This is accomplished by using a PW switching point TLV.

Figure 16 Pseudowire Switching: Example



In the above figure, the multisegment pseudowire is established between T-PE1 and T-PE2 with S-PE1 and S-PE2 as switching points. The pw-id 1 is between T-PE1 and S-PE1, pw-id 2 is between S-PE1 and S-PE2 and pw-id 3 is between S-PE2 and T-PE2.

Consider a packet traversal from T-PE1 to T-PE2:

1. T-PE1 sends label mapping message without a PW Switching Point TLV signal.
2. S-PE1 adds a PW Switching Point TLV signal with 4 sub-TLVs:
   a. description string
   b. pw-id 1
   c. S-PE1 IP address, which is the local address. This is the local router-id of the S-PE.
   d. T-PE1 IP address, which is the remote address
3. S-PE2 adds a PW Switching Point TLV signal with 3 sub-TLVs:
   a. description string
   b. pw-id 2
   c. S-PE2 IP address, which is the local address.
   d. No remote address because S-PE1 address is already present in the message as local IP address in the last TLV.
4. T-PE2 gets the label mapping message with 2 PW Switching TLVs.

Sometimes, you do not expose the information about previous S-PEs to the next S-PE for security reasons. By default, an S-PE appends its information to the PW Switching Point TLV signal. When "hiding" option is enabled on a PW segment using the "switching tlv hide" command, an S-PE sends a label mapping message without any PW Switching Point TLVs.

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.

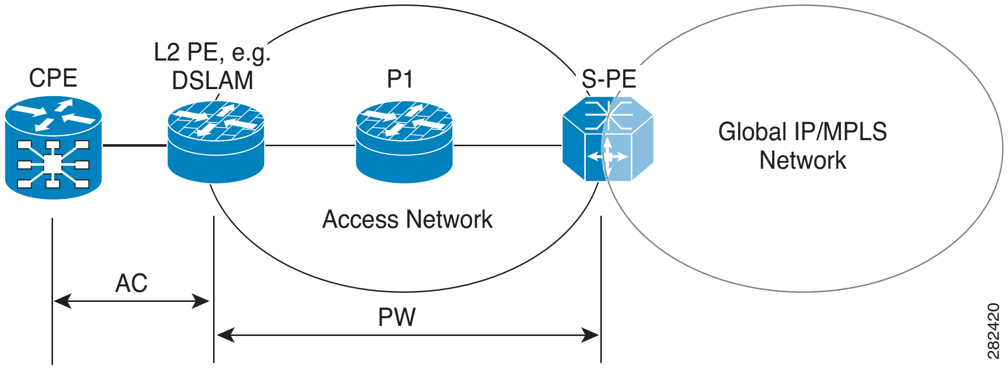
Pseudowire Headend

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

The PW headend (PWHE) feature provides a Layer 3 (L3) virtual interface representation of a PW on an service provider edge (PE), that allows the backhaul of customer packets over PWs and the application of L3 features, such as QoS (for example: policing and shaping), and access lists (ACLs) on customer packets on the PW.

The PWHE virtual interface originates as a PW on an access node (the Layer 2 PW feeder node) and terminates on a Layer 3 service instance, such as a VRF instance, on the service provider router. At the service PE, IP traffic on the PW (from a remote customer PE via the access network) is forwarded onto the IP/MPLS backbone and traffic from the IP/MPLS backbone, is forwarded onto the PWHE L3 interface towards the customer PE (via the access network).

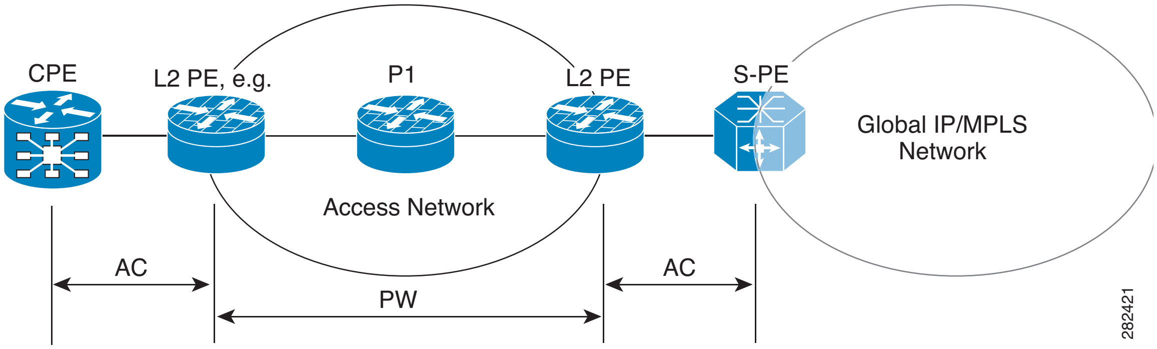
Figure 17 Example with PWHE



Note The PW is from L2 PE node to the Service PE (S-PE), but the L3 adjacency on each PWHE interface is configured between the service PE and the customer PE.

The PWHE feature allows you to replace a two node solution with a single node. Figure 18 illustrates a scenario wherein, without PWHE, an L2 PE node is required. The L2 PE node terminates the PW and connects to the service PE (from the L2 PE) via an attachment circuit (AC) that terminates as an L3 interface on the service PE.

Figure 18 Example without PWHE



PWHE Interfaces

The virtual circuit (VC) types supported for the PW are types 4, 5 and 11. The PWHE acts as broadcast interface with VC types 4 (VLAN tagged) and 5 (Ethernet port/Raw), whereas with VC type 11 (IP Interworking), the PWHE acts as a point-to-point interface.

eBGP Support on PWHE interfaces

To enable access CE to communicate with service PE, you need to configure eBGP on the PWHE interface. Enabling eBGP unblocks the path for all control packets (including eBGP) over PW-Ether (that is, for IPv4 and IPV6, and PW-IW interface for IPv4 only.)

IPv6 Unicast Support on PWHE

IPv6 Support is added by supporting the 6PE and 6VPE features. The Core network is either MPLS based or IP based. The IPv6 interfaces support both VC type 4 and 5.

Flow Aware Transport Pseudowire (FAT PW) Overview

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.

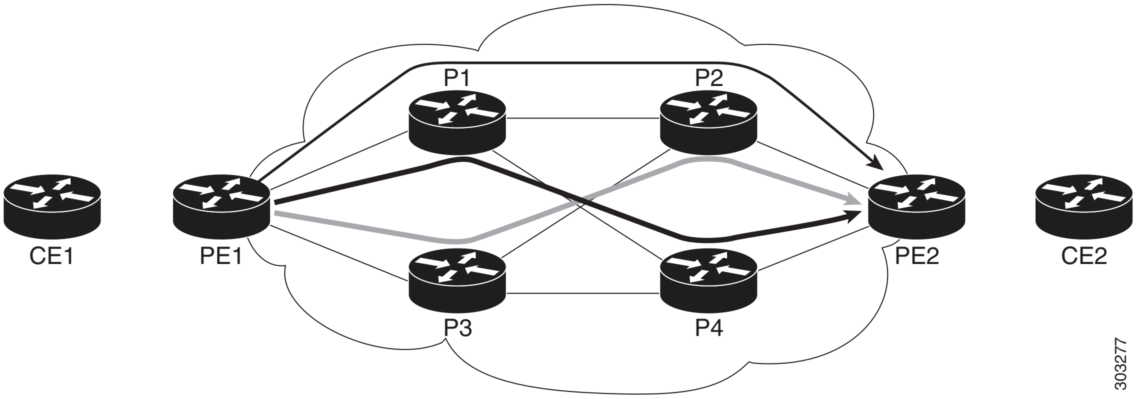
Flow-Aware Transport Pseudowires (FAT PW) provide the capability to identify individual flows within a pseudowire and provide routers the ability to use these flows to loadbalance traffic. FAT PWs are used to loadbalance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is 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 better traffic distribution across ECMP paths or link-bundled paths in the core.

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 the 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.

All core routers perform load balancing based on the flow-label in the FAT PW. Therefore, it is possible to distribute flows over ECMPs and link bundles.

Figure 19 shows a network with equal-cost multi-paths (ECMP).

Figure 19 Equal Cost Multi Path network



In Figure 19, traffic received from CE1 on PE1 load balances either P1 or P3, because the cost of links is equal. Further on P1, traffic flows from either P2 or P4. Similarly, P3 load balances either P2 or P4. The flow label helps to maximize load balancing on the P routers, throughout the network. The Ingress PE routers are responsible to add flow label whereas the Egress PE routers remove the flow label.

Pseudowire Grouping

When pseudowires (PW) are established, each PW is assigned a group ID that is common for all PWs created from the same physical port. Hence, when the physical port becomes non-functional or is deleted, L2VPN sends a single message to advertise the status change of all PWs belonging to the group. A single L2VPN signal thus avoids a lot of processing and loss in reactivity.

Note Pseudowire grouping is disabled by default.

How to Implement Virtual Private LAN Services

This section describes the tasks that are required to implement VPLS:

- Configuring a Bridge Domain, page VPC-101

- Configuring a Layer 2 Virtual Forwarding Instance, page VPC-116
- Configuring the MAC Address-related Parameters, page VPC-128
- Configuring Multisegment Pseudowire, page VPC-140
- Configuring Pseudowire Redundancy, page VPC-146
- Configuring Pseudowire Headend, page VPC-151
- Configuring Flow Aware Transport Pseudowire, page VPC-160
- Enabling Pseudowire Grouping, page VPC-164

Configuring a Bridge Domain

These topics describe how to configure a bridge domain:

- Creating a Bridge Domain, page VPC-101
- Configuring a Pseudowire, page VPC-103
- Associating Members with a Bridge Domain, page VPC-108
- Configuring Bridge Domain Parameters, page VPC-110
- Disabling a Bridge Domain, page VPC-112
- Blocking Unknown Unicast Flooding, page VPC-114

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/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 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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/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.

	Command or Action	Purpose

Step 6	**exit**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# exit`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Exits the current configuration mode.

Step 7	**neighbor** {*A.B.C.D*} {**pw-id** *value*}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/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	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-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 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** *number*
6. **backup neighbor** *ip-address* **pw-id** *number*
7. **end**

   or

   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**xconnect group** *group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group A`
	`RP/0/0/CPU0:router(config-l2vpn-xc)#`
		Enters the name of the cross-connect group.

Step 4	**p2p** *xconnect name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

Step 5	**neighbor** *ip-address* **pw-id** *number*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose

Step 6	**backup neighbor** *ip-address* **pw-id** *number*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# backup neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#`
		Configures the backup pseudowire for the point-to-point neighbor.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#end`
	or
	`RP/0/0/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.

Configuring Backup Disable Delay

The Backup Disable Delay function specifies the time for which the primary pseudowire in active state waits before it takes over for the backup pseudowire. Perform this task to configure a disable delay.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class name*
4. **backup disable delay** *seconds*
5. **exit**
6. **xconnect group** *group name*
7. **p2p** *xconnect name*
8. **neighbor** *ip-address* **pw-id** *number*
9. **pw-class** *class name*

10. **backup neighbor** *ip-address* **pw-id** *number*
11. **end**

    or

    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`pw-class`** *`class_1`*
	Example:
	`RP/0/RP0/CPU0:router(config-l2vpn)# pw-class class_1`
	`RP/0/RP0/CPU0:router(config-l2vpn-pwc)#`
		Configures the pseudowire class name.

Step 4	**`backup disable delay`** *`seconds`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20`
	`RP/0/0/CPU0:router(config-l2vpn-pwc)#`
		Specifies how long a backup pseudowire virtual circuit (VC) should wait before resuming operation after the primary pseudowire VC becomes nonfunctional.

Step 5	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# exit`
		Exits the pseudowire class submode.

Step 6	**`xconnect group`** *`group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group` *`A`*
	`RP/0/0/CPU0:router(config-l2vpn-xc)#`
		Enters the name of the cross-connect group.

Step 7	**`p2p`** *`xconnect name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p` *`rtrX_to_rtrY`*
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

	Command or Action	Purpose

Step 8	**neighbor** *ip-address* **pw-id** *number*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire segment for the cross-connect.

Step 9	**pw-class** *class_1*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class class_1`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)#`
		Configures the pseudowire class name.

Step 10	**backup neighbor** *ip-address* **pw-id** *number*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 1.1.1.1 pw-id 2`
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#`
		Configures the backup pseudowire for the point-to-point neighbor.

Step 11	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#end`
	or
	`RP/0/0/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.

Associating Members with a Bridge Domain

After a bridge domain is created, perform this task to assign interfaces to the bridge domain. The following 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. **static-mac-address** {*MAC-address*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0`
	`RP/0/0/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.

	Command or Action	Purpose

Step 6	**`static-mac-address`** {*`MAC-address`*}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# 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 7	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-ac)# 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 Bridge Domain Parameters

To configure the bridge domain parameters, associate the following 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—Enables or disables flooding on the bridge domain. By default, flooding is enabled.

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters l2vpn configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# flooding disable`	Configures flooding for traffic at the bridge domain level or at the bridge port level.

	Command or Action	Purpose

Step 6	**`mtu`** *`bytes`*
	Example:
	`RP/0/0/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	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/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.

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

	Command or Action	Purpose
Step 5	**shutdown** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`	Shuts down a bridge domain to bring the bridge and all attachment circuits and pseudowires under it to admin down state.

Step 6	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/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.

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. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`bridge group`** *`bridge-group-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

	Command or Action	Purpose

Step 5	**`flooding unknown-unicast disable`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# flooding unknown-unicast disable`
		Disables flooding of unknown unicast traffic at the bridge domain level.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# end`
	or
	`RP/0/0/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 a Layer 2 Virtual Forwarding Instance

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

- Adding the Virtual Forwarding Instance Under the Bridge Domain, page VPC-117
- Associating Pseudowires with the Virtual Forwarding Instance, page VPC-118
- Associating a Virtual Forwarding Instance to a Bridge Domain, page VPC-120
- Attaching Pseudowire Classes to Pseudowires, page VPC-122
- Configuring Any Transport over Multiprotocol Pseudowires By Using Static Labels, page VPC-124
- Disabling a Virtual Forwarding Instance, page VPC-126

Adding the Virtual Forwarding Instance Under the Bridge Domain

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`bridge group`** *`bridge group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

	Command or Action	Purpose

Step 5	**vfi** {*vfi name*}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/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	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-vpn)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-vpn)# 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.

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`bridge group`** *`bridge group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)#`
		Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.

	Command or Action	Purpose

Step 6	**`neighbor`** *`A.B.C.D`* `{`**`pw-id`** *`value`*`}`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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 7	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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.

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. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`bridge group`** *`bridge group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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.

	Command or Action	Purpose

Step 7	**`static-mac-address`** {*`MAC address`*}
	Example:
	`RP/0/0/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	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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.

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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`bridge group`** *`bridge group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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.

	Command or Action	Purpose

Step 7	**pw-class** {*class name*}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada`
		Configures the pseudowire class template name to use for the pseudowire.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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 Any Transport over Multiprotocol Pseudowires By 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. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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.

	Command or Action	Purpose

Step 7	**mpls static label** {**local** *value*} {**remote** *value*}
	Example:
	`RP/0/0/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 access pseudowire configuration. You can set the local and remote pseudowire labels.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi-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.

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd)# vfi v1`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown`	Disables the virtual forwarding interface (VFI).

	Command or Action	Purpose

Step 7	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-vfi)# 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 l2vpn bridge-domain`** `[`**`detail`**`]`
	`Example:`
	`RP/0/0/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:

- Configuring the MAC Address Source-based Learning, page VPC-129
- Disabling the MAC Address Withdrawal, page VPC-131
- Configuring the MAC Address Limit, page VPC-133
- Configuring the MAC Address Aging, page VPC-135
- Disabling MAC Flush at the Bridge Port Level, page VPC-138

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

	Command or Action	Purpose

Step 6	**`learning disable`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# learning disable`
		Overrides the MAC learning configuration of a parent bridge or sets the MAC learning configuration of a bridge.

Step 7	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# 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 l2vpn bridge-domain`** [**`detail`**] **Example:** `RP/0/0/CPU0:router# show l2vpn bridge-domain detail`	Displays the details that the MAC address source-based learning is disabled on the bridge.

Disabling the MAC Address Withdrawal

Perform this task to disable the MAC address withdrawal for a specified bridge domain.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **withdraw** { **access-pw disable | disable** }
7. **end**
   or
   commit
8. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td><b>Step 6</b></td><td><b>withdraw { access-pw disable | disable }</b>

<b>Example:</b>
RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# withdraw access-pw disable</td><td>Disables the MAC address withdrawal for the specified bridge domain.
<b>Note</b> Mac address withdrawal is generated when the access pseudowire is not operational.</td></tr>
<tr><td><b>Step 7</b></td><td><b>end</b>
or
<b>commit</b>

<b>Example:</b>
RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end
or
RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# commit</td><td>Saves configuration changes.
• When you issue the <b>end</b> command, the system prompts you to commit changes:
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
– Entering <b>yes</b> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
– Entering <b>no</b> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
– Entering <b>cancel</b> leaves the router in the current configuration session without exiting or committing the configuration changes.
• Use the <b>commit</b> command to save the configuration changes to the running configuration file and remain within the configuration session.</td></tr>
<tr><td><b>Step 8</b></td><td><b>show l2vpn bridge-domain</b> [<b>detail</b>]

<b>Example:</b>
P/0/0/CPU0:router# show l2vpn bridge-domain detail</td><td>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.</td></tr>
</table>

The following sample output shows the MAC address withdrawal fields:

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

Bridge group: siva_group, bridge-domain: siva_bd, id: 0, state: up, ShgId: 0, MSTi: 0
  MAC Learning: enabled
  MAC withdraw: enabled
  Flooding:
    Broadcast & Multicast: enabled
    Unknown Unicast: enabled
  MAC address aging time: 300 s Type: inactivity
  MAC address limit: 4000, Action: none, Notification: syslog
  MAC limit reached: no
  Security: disabled
  DHCPv4 Snooping: disabled
  MTU: 1500
  MAC Filter:  Static MAC addresses:
  ACs: 1 (1 up), VFIs: 1, PWs: 2 (1 up)
```

```
List of ACs:
  AC: GigabitEthernet0/4/0/1, state is up
    Type Ethernet
    MTU 1500; XC ID 0x5000001; interworking none; MSTi 0 (unprotected)
    MAC Learning: enabled
    MAC withdraw: disabled
    Flooding:
      Broadcast & Multicast: enabled
      Unknown Unicast: enabled
    MAC address aging time: 300 s Type: inactivity
    MAC address limit: 4000, Action: none, Notification: syslog
    MAC limit reached: no
    Security: disabled
    DHCPv4 Snooping: disabled
    Static MAC addresses:
    Statistics:
      packet totals: receive 6,send 0
      byte totals: receive 360,send 4
List of Access PWs:
List of VFIs:
  VFI siva_vfi
    PW: neighbor 1.1.1.1, PW ID 1, state is down ( local ready )
      PW class not set, XC ID 0xff000001
      Encapsulation MPLS, protocol LDP
      PW type Ethernet, control word enabled, interworking none
      PW backup disable delay 0 sec
      Sequencing not set
              MPLS         Local                          Remote
        ------------ ------------------------------ -------------------------
             Label          30005                          unknown
             Group ID     0x0                              0x0
             Interface    siva/vfi                         unknown
             MTU          1500                             unknown
             Control word enabled                     unknown
             PW type      Ethernet                      unknown
        ------------ ------------------------------ -------------------------
      Create time: 19/11/2007 15:20:14 (00:25:25 ago)
      Last time status changed: 19/11/2007 15:44:00 (00:01:39 ago)
      MAC withdraw message: send 0 receive 0
```

Configuring the MAC Address Limit

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

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **bridge group** *bridge group name*
4. **bridge-domain** *bridge-domain name*
5. **mac**
6. **limit**
7. **maximum** {*value*}
8. **action** {**flood** | **no-flood** | **shutdown**}
9. **notification** {**both** | **none** | **trap**}

10. **end**
or
commit

11. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge group name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**mac**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6	**limit**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# limit`
	`RP/0/0/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 7	**maximum** {*value*}
	Example:
	`RP/0/0/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.

	Command or Action	Purpose

Step 8	**action** {**flood** | **no-flood** | **shutdown**}
	Example:
	`RP/0/0/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 9	**notification** {**both** | **none** | **trap**}
	Example:
	`RP/0/0/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 10	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# 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 11	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/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. **type** {**absolute** | **inactivity**}
9. **end**
or
commit
10. **show l2vpn bridge-domain** [**detail**]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**`bridge group`** *`bridge group name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

Step 5	**`mac`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters L2VPN bridge group bridge domain MAC configuration mode.

Step 6	**`aging`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# aging`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#`
		Enters the MAC aging configuration submode to set the aging parameters such as time and type.

	Command or Action	Purpose

Step 7	**time** {*seconds*}
	Example:
	`RP/0/0/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	**type** {**absolute** | **inactivity**}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# type absolute`
		Configures the type for MAC address aging.
		• Use the **absolute** keyword to configure the absolute aging type.
		• Use the **inactivity** keyword to configure the inactivity aging type.

Step 9	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# 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 10	**show l2vpn bridge-domain** [**detail**] **Example:** `RP/0/0/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, bridge port or access pseudowire levels. 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. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**l2vpn**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

Step 3	**bridge group** *bridge-group-name*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# bridge group csco`
	`RP/0/0/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/0/0/CPU0:router(config-l2vpn-bg)# bridge-domain abc`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)#`
		Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.

Step 5	**mac**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd)# mac`
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)#`
		Enters l2vpn bridge group bridge domain MAC configuration mode.

	Command or Action	Purpose
Step 6	**port-down flush disable** **Example:** `RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# port-down flush disable`	Disables MAC flush when the bridge port becomes nonfunctional.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-bg-bd-mac)# 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 Multisegment Pseudowire

This section describes these tasks:

- Provisioning a Multisegment Pseudowire Configuration, page VPC-140
- Provisioning a Global Multisegment Pseudowire Description, page VPC-142
- Provisioning a Cross-Connect Description, page VPC-143
- Provisioning Switching Point TLV Security, page VPC-144
- Enabling Multisegment Pseudowires, page VPC-145

Provisioning a Multisegment Pseudowire Configuration

Configure a multisegment pseudowire as a point-to-point (p2p) cross-connect. For more information refer, Figure 15 on page 96. Here, the **xconnect group** item corresponds to the MPLS/IP. The **neighbor** item corresponds to the destination PE node with its IP address and the **pw-id**.

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

	Command	Purpose

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.

	Command	Purpose
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** The psuedowire configuration is done on an S-PE node.
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. **Note** The psuedowire configuration is done on an S-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. For more information refer, Figure 16 on page 97.

Each multisegment pseudowire can have its own description. For instructions, see the "Provisioning a Cross-Connect Description" section on page 143. If it does not have one, this global description is used.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **description** *value*
4. **commit**

DETAILED STEPS

	Command	Purpose
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	**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. **Note** The psuedowire configuration is done on all S-PE nodes.
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

	Command	Purpose
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 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.

	Command	Purpose
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" section on page 140. **Note** The psuedowire configuration is done on all S-PE nodes.
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".

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

	Command	Purpose
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.

	Command	Purpose

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.
		Note The psuedowire configuration is done on all S-PE nodes.

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

	Command	Purpose

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 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 comes back up.

These topics describe how to configure pseudowire redundancy:

- Configuring a Backup Pseudowire, page VPC-146
- Configuring Point-to-Point Pseudowire Redundancy, page VPC-149
- Forcing a Manual Switchover to the Backup Pseudowire, page VPC-151

Configuring a Backup Pseudowire

Perform this task to configure a backup pseudowire for a point-to-point neighbor.

Note When you reprovision a primary pseudowire, traffic resumes in two seconds. However, when you reprovision a backup pseudowire, traffic will resume after a delay of 45 to 60 seconds. This is the expected behavior.

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. **backup** {**neighbor** *A.B.C.D*} {**pw-id** *value*}
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

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	**`xconnect group`** *`group-name`*
	Example:
	`RP/O/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 xc1`
	`RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)#`
		Enters a name for the point-to-point cross-connect.

Step 5	**`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`
		Configures the pseudowire segment for the cross-connect.

	Command or Action	Purpose

Step 6	**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 cross-connect. The IP address argument (*A.B.C.D*) 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 7	**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.

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

	Command or Action	Purpose

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.

	Command or Action	Purpose

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/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`
	`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.

	Command or Action	Purpose

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 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 Pseudowire Headend

The PWHE is created by configuring interface pw-ether or pw-iw. For the PWHE to be functional, the xconnect has to be configured completely. Configuring other layer 3 (L3) parameters, such as VRF and IP addresses, are optional for the PWHE to be functional. However, the L3 features are required for the layer 3 services to be operational; that is, for PW L3 termination.

This section describes these topics:

- PWHE Configuration Restrictions
- Configuring PWHE Interfaces
- Configuring PWHE Interface Parameters
- Configuring PWHE Crossconnect

PWHE Configuration Restrictions

These are the configuration restrictions for PWHE:

- Up to 3600 PWHE interfaces (a combination of pw-ether and pw-iw).
- Up to eight interface lists per peer.
- Up to four L3 links per interface list.
- VLAN ID (tag-impose) can be configured only in xconnects which have pw-ether interfaces.
- VLAN ID (tag-impose) can only be configured under VC type 4 pw-ether interfaces.
- Interface lists can accept POS, GigabitEthernet, TenGigabitEthernet; other interfaces are rejected.
- No support for features such as pseudowire redundancy, preferred path, local switching or L2TP for xconnects configured with PWHE.
- Ethernet and VLAN transport modes are not allowed for pw-iw xconnects.
- Address family, Cisco Discovery Protocol (CDP) and MPLS configurations are not allowed on PWHE interfaces.
- IPv6 configuration is not allowed under pw-iw interfaces.

Configuring PWHE Interfaces

Perform this task to configure PWHE interfaces.

Summary Steps

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

Detailed Steps

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/0/CPU0:router# configure`
	`RP/0/0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	`interface pw-ether` *`id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface pw-ether <id>`
		Configures the PWHE interface and enters the interface configuration mode.

Step 3	`attach generic-interface-list` *`interface_list_name`*
	Example:
	`RP/0/0/CPU0:router(config-if)# attach generic-interface-list interfacelist1`
		Attaches the interface to a specified interface list.

Step 4	`end`
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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.

Restrictions for Configuring PWHE Interfaces

These are the restrictions for configuring PWHE interfaces:

- Neighbor and pw-ID pair must be unique in L2VPN.
- pw-ether interfaces have to be VC type 4 or 5.
- pw-iw interfaces cannot have IPv6 address because IPv6 is not supported on pw-iw (VC type 11). The VC type is set to type 11 if AC is pw-iw even when interworking ipv4 is not configured.
- The VLAN ID is allowed only if VC type is 4.

- MPLS protocols (MPLS-TE, LDP, RSVP) cannot be configured on PW-HE.
- No interface list configuration is accepted on non-PWHE platforms.

Configuring PWHE Interface Parameters

Perform this task to configure PWHE interface parameters.

Summary Steps

1. **configure**
2. **interface pw-ether** *id*
3. **attach generic-interface-list** *interface_list_name*
4. **l2overhead** *bytes*
5. **load-interval** *seconds*
6. **dampening** *decay-life*
7. **logging events link-status**
8. **mac-address** *MAC address*
9. **mtu** *interface_MTU*
10. **bandwidth** *bandwidth*
11. **end**
    or
    commit

Detailed Steps

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router#` **configure** `RP/0/0/CPU0:router(config)#`	Enters global configuration mode.
Step 2	**interface pw-ether** *id* **Example:** `RP/0/0/CPU0:router(config)# interface pw-ether <id>`	Configures the PWHE interface and enters the interface configuration mode.
Step 3	**attach generic-interface-list** *interface_list_name* **Example:** `RP/0/0/CPU0:router(config-if)# attach generic-interface-list interfacelist1`	Attaches the interface to a specified interface list.
Step 4	**l2overhead** *bytes* **Example:** `RP/0/0/CPU0:router(config-if)#l2overhead 20`	Sets layer 2 overhead size.
Step 5	**load-interval** *seconds* **Example:** `RP/0/0/CPU0:router(config-if)#load-interval 90`	Specifies interval, in seconds, for load calculation for an interface. The number of seconds: • Can be set to 0 [0 disables load calculation] • If not 0, interval must be specified in multiples of 30 between 30 and 600.
Step 6	**dampening** *decay-life* **Example:** `RP/0/0/CPU0:router(config-if)#dampening 10`	Configures state dampening on the given interface (in minutes).
Step 7	**logging events link-status** **Example:** `RP/0/0/CPU0:router(config-if)#logging events link-status`	Configures per interface logging.
Step 8	**mac-address** *MAC address* **Example:** `RP/0/0/CPU0:router(config-if)#mac-address aaaa.bbbb.cccc`	Sets the MAC address (xxxx.xxxx.xxxx) on an interface.

	Command or Action	Purpose
Step 9	**mtu** *interface_MTU* **Example:** `RP/0/0/CPU0:router(config-if)#mtu 128`	Sets the MTU on an interface.
Step 10	**bandwidth** *bandwidth* **Example:** `RP/0/0/CPU0:router(config-if)#bandwidth 3987`	Sets the bandwidth of an interface.

Step 11	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 Pseudowire Source Address

Source address is configured under pseudowire class with encapsulation set to MPLS. This enables flexible LDP target to support Rx pindown. Perform this task to configure the source IPv4 address.

Summary Steps

1. **configure**
2. **l2vpn**
3. **pw-class** *class-name*
4. **encapsulation mpls**
5. **ipv4 source** *A.B.C.D*
6. **end**
   or
   commit

Detailed Steps

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/0/CPU0:router# **configure** RP/0/0/CPU0:router(config)#	Enters global configuration mode.
Step 2	**l2vpn** **Example:** RP/0/0/CPU0:router(config)# l2vpn	Enters Layer 2 VPN configuration mode.
Step 3	**pw-class** *class-name* **Example:** RP/0/0/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/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls	Sets pseudowire encapsulation to MPLS.
Step 5	**ipv4 source** *A.B.C.D* **Example:** RP/0/0/CPU0:router(config-l2vpn-pwc-mpls)# ipv4 source w-ether 100	Sets the local source IPv4 address.
Step 6	**end** or **commit** **Example:** RP/0/0/CPU0:router(config-l2vpn-pwc-mpls)# end or RP/0/0/CPU0:router(config-l2vpn-pwc-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.

Configuring PWHE Crossconnect

Perform this task to configure PWHE crossconnects.

Summary Steps

1. **configure**
2. **l2vpn**
3. **xconnect group** *group-name*
4. **p2p** *xconnect-name*
5. **interface pw-ether** *id*
6. **neighbor** *A.B.C.D* **pw-id** *value*
7. **pw-class** *class-name*
8. **end**
   or
   commit

Detailed Steps

	Command or Action	Purpose

Step 1	**configure**
	Example:
	`RP/0/0/CPU0:router# configure`
	`RP/0/0/CPU0:router(config)#`
		Enters global configuration mode.

Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters Layer 2 VPN configuration mode.
Step 3	**xconnect group** *group-name* **Example:** `RP/0/0/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/0/0/CPU0:router(config-l2vpn-xc)# p2p ms-pw1`	Enters P2P configuration submode.
Step 5	**interface pw-ether** *id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface pw-ether 100`	Configures the PWHE interface.
Step 6	**neighbor** *A.B.C.D* **pw-id** *value* **Example:** `RP/0/0/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.

	Command or Action	Purpose
Step 7	**pw-class** *class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class dynamic_mpls`	Enters pseudowire class submode, allowing you to define a pseudowire class template.

Step 8	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 Flow Aware Transport Pseudowire

This section provides information on

- Enabling Load Balancing with ECMP and FAT PW for VPWS

Enabling Load Balancing with ECMP and FAT PW for VPWS

Perform this task to enable load balancing with ECMP and FAT PW for VPWS.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** {*name*}
4. **encapsulation mpls**
5. **load-balancing flow-label** {**both** | **receive** | **transmit**} [**static**]
6. **exit**
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** {*name*}
12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters the configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enters L2VPN configuration mode.

Step 3	**pw-class** {*name*}
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# pw-class path1`
		Configures the pseudowire class template name to use for the pseudowire.

Step 4	**encapsulation mpls** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation mpls`	Configures the pseudowire encapsulation to MPLS.

	Command or Action	Purpose
Step 5	**load-balancing flow-label** {**both** | **receive** | **transmit**} [**static**] **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-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/0/0/CPU0:router(config-l2vpn-pwc-encap-mpls)#exit`	Exits the pseudowire encapsulation submode and returns the router to the parent configuration mode.
Step 7	**xconnect group** *group-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp1`	Specifies the name of the cross-connect group.
Step 8	**p2p** *xconnect-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p vlan1`	Specifies the name of the point-to-point cross-connect
Step 9	**interface type** *interface-path-id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1`	Specifies the interface type and instance.
Step 10	**neighbor** *A.B.C.D* **pw-id** *pseudowire-id* **Example:** `RP/0/0/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.

	Command or Action	Purpose
Step 11	`pw-class` *class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class path1`	Associates the pseudowire class with this pseudowire.

Step 12	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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.

Enabling Pseudowire Grouping

Perform this task to enable pseudowire grouping.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
	`RP/0/0/CPU0:router(config-l2vpn)#`
		Enters L2VPN configuration mode.

	Command or Action	Purpose

Step 3	**`pw-grouping`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# pw-grouping`
		Enables pseudowire grouping

Step 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# end`
	or
	`RP/0/0/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.

Configuration Examples for Virtual Private LAN Services

This section includes the following configuration examples:

- Virtual Private LAN Services Configuration for Provider Edge-to-Provider Edge: Example, page VPC-166
- Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example, page VPC-167
- Configuring Backup Disable Delay: Example, page VPC-167
- Blocking Unknown Unicast Flooding: Example, page VPC-168
- Disabling MAC Flush: Examples, page VPC-168
- H-VPLS with QinQ or QinAny: Examples, page VPC-169
- H-VPLS with Access-PWs: Examples, page VPC-170
- Pseudowires: Examples, page VPC-170
- Configuring Multisegment Pseudowires: Examples, page VPC-174
- Configuring Pseudowire Redundancy: Examples, page VPC-176

- Configuring Pseudowire Headend: Example, page VPC-178
- Configuring Flow Aware Transport Pseudowire: Example, page VPC-182
- Enabling Pseudowire Grouping: Example, page VPC-182

Virtual Private LAN 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 VPLS provider edge (PE) nodes.

The following configuration example shows how to configure PE 1:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE1-VPLS-A
    GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 2.2.2.2 pw-id 1---PW1
     neighbor 3.3.3.3 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 1.1.1.1 255.255.255.25
  commit
```

The following configuration example shows how to configure PE 2:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE2-VPLS-A
    interface GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 1.1.1.1 pw-id 1---PW1
     neighbor 3.3.3.3 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 2.2.2.2 255.255.255.25
  commit
```

The following configuration example shows how to configure PE 3:

```
configure
 l2vpn
  bridge group 1
   bridge-domain PE3-VPLS-A
    interface GigabitEthernet0/0---AC
     exit
    vfi 1
     neighbor 1.1.1.1 pw-id 1---PW1
     neighbor 2.2.2.2 pw-id 1---PW2
     !
   !
 interface loopback 0
  ipv4 address 3.3.3.3 255.255.255.25
  commit
```

Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example

The following configuration shows how to configure VPLS for a PE-to-CE nodes:

```
configure
 interface GigabitEthernet0/0
  l2transport---AC interface
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end

configure
 interface GigabitEthernet0/0
  l2transport
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
  end

configure
 interface GigabitEthernet0/0
  l2transport
   exit
  no ipv4 address
  no ipv4 directed-broadcast
  negotiation auto
  no cdp enable
```

Configuring Backup Disable Delay: Example

The following example shows how a backup delay is configured for point-to-point PW where the backup disable delay is 50 seconds:

```
l2vpn
pw-class class_1
backup disable delay 20
exit
xconnect group_A
p2p rtrX_to_rtrY
neighbor 1.1.1.1 pw-id 2
pw-class class_1
backup neighbor 2.2.2.2 pw- id 5
commit
```

The following example shows how a backup delay is configured for point-to-point PW where the backup disable delay is never:

```
l2vpn
pw-class class_1
backup disable  never
exit
```

```
xconnect group_A
p2p rtrX_to_rtrY
     neighbor 1.1.1.1 pw-id 2
pw-class class_1
     backup neighbor 2.2.2.2 pw-id 5
commit
```

Blocking Unknown Unicast Flooding: Example

Unknown-unicast flooding can be blocked at the following levels:

- bridge domain
- bridge port (attachment circuit (AC))
- access pseudowire (PW)

The following 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
```

The following example shows how to block unknown-unicast flooding at the bridge port level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    interface POS 0/1/0/1
    flooding unknown-unicast disable
  end
```

The following 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 the following 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 POS 0/1/0/1
    mac
    port-down flush disable
  end
```

The following example shows how to disable the MAC flush at the access pseudowire level:

```
configure
  l2vpn
    bridge-group group1
    bridge-domain domain1
    neighbor 10.1.1.1 pw-id 1000
    mac
    port-down flush disable
   end
```

H-VPLS with QinQ or QinAny: Examples

This example shows the QinQ or QinAny AC type in the output of the **show l2vpn forwarding bridge-domain hardware ingress/egress** command:

```
INGRESS AC [version, state]: [1, BOUND]

        Xconnect-ID: [15] TCAM-Key: (UIDB:0x4 O-vlan:2 I-vlan:2 Ether-Type:0x8100)
        HW: 0x34001000 0x0118000f 0x1011801c 0xc7ff5100
        SW: 0x34001000 0x0118000f 0x1011801c 0xc7ff5100

        Service type: 7 (bridging pmp QinQ)
        Entry type: 1 (fwd)
        Bridge_ID : 0
        ACL_ID : 4096
        Xconnect_ID : 0x118000f
        SplitHorizonGroup_ID : 0
        Rewrite supported: 0 (No)
        PW_mode: 0 (vc-type 5)
        AC-type: 1 (qinq-mode)
        Interface handle: 0x11801c
        Ingress AC stats: 0x7ff51

   EGRESS AC [version, state]: [1, BOUND]

        Xconnect-ID: [15] TLU2-entry-addr: [0x200a00f]
        HW: 0x8018b000 0x0000000f 0x00004002 0xfb748000
        SW: 0x8018b000 0x0000000f 0x00004002 0xfb748000

        Entry status: 1 (Fwd)
        AC_type: 1 (qinq-mode)
        Outer-vlan: 2
        Inner-vlan: 2
        Outer Ether Type: 0 (dot1q)
        AC_mtu: 1580
        Adjacency_type: 3
        Default EgressQ (SharqQ): 15
        PW mode: 0 (vc-type 5)
        Rewrite supported: 0 (No)
        Control-word supported: 0 (No)
        Egress AC stats: 0x7dba4
```

H-VPLS with Access-PWs: Examples

This example shows the PW type in the output of the **show l2vpn forwarding bridge-domain hardware ingress/egress** command:

```
Ingress:
  INGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: Access PW
        XID: 127/15/CPU0 : 1 (0xfff80001)
        Bridge ID: 0, Split Horizon ID: 0
        VC label: 16010

  INGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: VFI(Core) PW
        XID: 127/15/CPU0 : 2 (0xfff80002)
        Bridge ID: 0, Split Horizon ID: 1
        VC label: 16007

Egress:
        OIF[1] seg_type: Access PW xid: 0xfff80001 ecd_ptr: 0x500a
        TLU RESULT tlu_addr: 0x200bc00 ch: 2 seg_type: 0
        HW: 0x0000500a 0x00000000 0xfff80001 0x03e8a000
        SW: 0x0000500a 0x00000000 0xfff80001 0x03e8a000
        SHG: 0
        XID: 0xfff80001
…
        OIF[2] seg_type: VFI(Core) PW xid: 0xfff80002 ecd_ptr: 0x500f
        TLU RESULT tlu_addr: 0x3000601 ch: 3 seg_type: 0
        HW: 0x0100500f 0x00000000 0xfff80002 0x03e87000
        SW: 0x0100500f 0x00000000 0xfff80002 0x03e87000
        SHG: 1
        XID: 0xfff80002
…
  EGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: Access PW
        XID: 127/15/CPU0 : 1 (0xfff80001)
        Bridge ID: 0, Split Horizon ID: 0
        VC label: 16010
…
  EGRESS BRIDGE PORT [version, state]: [1, BOUND]
        Bridge Port Type: VFI(Core) PW
        XID: 127/15/CPU0 : 2 (0xfff80002)
        Bridge ID: 0, Split Horizon ID: 1
        VC label: 16007
```

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
```

Configuring Multisegment Pseudowires: Examples

This example shows how to configure a multisegment pseudowire:

```
configure
   l2vpn
      xconnect group MS-PW1
      p2p ms-pw1
      neighbor 10.165.200.25 pw-id 100
```

```
        pw-class dynamic_mpls
        exit
        neighbor 10.165.202.158 pw-id 300
        pw-class dynamic_mpls
    end
```

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)
```

show l2vpn xconnect summary

```
RP/0/RSP0/CPU0:router# show l2vpn xconnect summary
Number of groups: 1
Number of xconnects: 1
  Up: 1  Down: 0  Unresolved: 0
  AC-PW: 0  AC-AC: 0  PW-PW: 1
Number of Admin Down segments: 0
```

Configuring Pseudowire Redundancy: Examples

This example shows how to configure a backup pseudowire for a point-to-point neighbor:

```
configure
    l2vpn
        xconnect group A
        p2p xc1
        neighbor 10.1.1.2 pw-id 2
        backup neighbor 10.2.2.2 pw-id 5
    end
```

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, SB = Standby
```

```
XConnect                   Segment 1                  Segment 2
Group      Name       ST   Description           ST   Description            ST
------------------------   -------------------------   -------------------------
g1         pw2        UP   Gi0/2/0/0.2           UP   110.110.110.110 2      UP
                                                       Backup
                                                       130.130.130.130 2      SB
--------------------------------------------------------------------------------
```

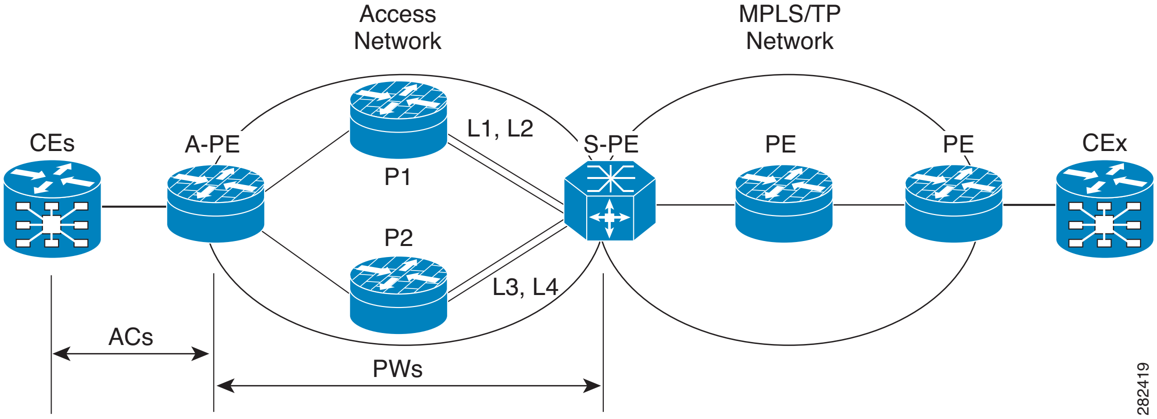
show l2vpn xconnect detail

```
RP/0/RSP0/CPU0:router# show l2vpn xconnect detail
Group MS-PW1, XC ms-pw1, state is up; Interworking none
  PW: neighbor 10.165.200.225, 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 PW Switching TLVs (Label Mapping message):
      None
    Incoming Status (PW Status TLV and accompanying PW Switching TLV):
      Status code: 0x0 (no fault) in Notification message
    Outgoing PW Switching TLVs (Label Mapping message):
      Local IP Address: 10.165.200.254 , Remote IP address: 10.165.202.158 , PW ID: 300
      Description: S-PE1 MS-PW between 10.165.200.225 and 10.165.202.158
    Outgoing Status (PW Status TLV and accompanying PW Switching TLV):
      Status code: 0x0 (no fault) in Notification message
      Local IP Address: 10.165.200.254
    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)
    Statistics:
      packet totals: receive 0
      byte totals: receive 0
  PW: neighbor 10.165.202.158 , 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 Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Create time: 04/02/2009 19:28:59 (00:21:04 ago)
    Last time status changed: 04/02/2009 19:46:12 (00:03:51 ago)
    MAC withdraw message: send 0 receive 0
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
Backup PW:
  PW: neighbor 130.130.130.130, PW ID 2, state is standby ( all ready )
    Backup for neighbor 110.110.110.110 PW ID 2 ( inactive )
    PW class dynamic_mpls, XC ID 0x3000002
    Encapsulation MPLS, protocol LDP
    PW type Ethernet, control word enabled, interworking none
    Sequencing not set
    PW Status TLV in use
      MPLS         Local                          Remote
      ------------ ------------------------------ -----------------------------
      Label        16001                          16002
      Group ID     0x3000200                      0x4
      Interface    GigabitEthernet0/2/0/0.2       3
      MTU          1500                           1500
      Control word enabled                        enabled
      PW type      Ethernet                       Ethernet
      VCCV CV type 0x2                            0x2
                   (LSP ping verification)        (LSP ping verification)
      VCCV CC type 0x7                            0x7
                    (control word)                 (control word)
                   (router alert label)           (router alert label)
                   (TTL expiry)                   (TTL expiry)
      ------------ ------------------------------ -----------------------------
Incoming Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Outgoing Status (PW Status TLV):
      Status code: 0x0 (Up) in Notification message
    Create time: 04/02/2009 19:28:59 (00:21:04 ago)
    Last time status changed: 04/02/2009 19:46:12 (00:03:51 ago)
    MAC withdraw message: send 0 receive 0
    Statistics:
      packets: received 0, sent 0
      bytes: received 0, sent 0
```

Configuring Pseudowire Headend: Example

This section provides an example of pseudowire headend configuration.

Figure 20 PWHE Configuration Example



Consider the topology in the above figure.

1. There are many customer edge routers (CEs) connected to a A-PE (each CE is connected using 1 link).
2. There are two P routers between A-PE an S-PE in the access network.
3. S-PE is connected by two links to P1—links L1 and L2 (on two separate linecards on P1 and S-PE); for example, Gig0/1/0/0 and Gig0/2/0/0 respectively.
4. S-PE is connected by two links to P2—L3 and L4 (on two separate linecards on P2 and S-PE); for example, Gig0/1/0/1 and Gig0/2/0/1 respectively.
5. For each CE-APE link, a xconnect (AC-PW) is configured on the A-PE. The PWs are connected to S-PE; some PWs are connected to [L1 (Gig0/1/0/0), L4 (Gig0/2/0/1)] and others through [L2 (Gig0/1/0/1), L3 (Gig0/2/0/0)].
6. A-PE uses router-id 100.100.100.100 for routing and PW signaling.
7. The two router-ids on S-PE used for PW signaling are 111.111.111.111 and 112.112.112.112 (for Rx pin-down). 110.110.110.110 is the router-id assigned for routing.

CE Configuration

Consider two CEs connected using GigabitEthernet0/3/0/0 (CE1 and A-PE) and GigabitEthernet0/3/0/1 (CE2 and A-PE).

At CE1:

```
interface Gig0/3/0/0
 ipv4 address 10.1.1.1/24
router static
 address-family ipv4 unicast
  110.110.110.110 Gig0/3/0/0
  A.B.C.D/N 110.110.110.110
```

At CE2:

```
interface Gig0/3/0/1
 ipv4 address 10.1.2.1/24
router static
 address-family ipv4 unicast
  110.110.110.110 Gig0/3/0/1
  A.B.C.D/N 110.110.110.110
```

A-PE Configuration

At A-PE, one xconnect is configured for each CE connection. Here, CE connections are L2 links, which are in xconnects. Each xconnect has a pseudowire connected to S-PE, though connected to different neighbor addresses, depending on where the pseudowire is to be pin downed: [L1, L4] or [L2, L3].

```
interface Gig0/3/0/0
 l2transport
interface Gig0/3/0/1
 l2transport

l2vpn
 xconnect group pwhe
  p2p pwhe_spe_1
   interface Gig0/3/0/0
   neighbor 111.111.111.111 pw-id 1
  p2p pwhe_spe_2
   interface Gig0/3/0/1
   neighbor 112.112.112.112 pw-id 2
```

P Router Configuration

Static routes are required on P routers for Rx pindown on S-PE to force PWs configured with a specific address to be transported over certain links.

At P1:

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

At P2:

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

S-PE Configuration

At S-PE, two PWHE interfaces (one for each PW) is configured, and each uses a different interface list for Tx pin-down. (This must match the static configuration at P routers for Rx pin-down). Each PWHE has the PW connected to A-PE (The pw-id must match the pw-id at A-PE.)

```
generic-interface-list il1
 interface gig0/1/0/0
 interface gig0/2/0/0
generic-interface-list il2
 interface gig0/1/0/1
 interface gig0/2/0/1

interface pw-ether1
 ipv4 address 10.1.1.2/24
 attach generic-interface-list il1
interface pw-ether2
 ipv4 address 10.1.2.2/24
 attach generic-interface-list il2

l2vpn
 xconnect group pwhe
  p2p pwhe1
   interface pw-ether1
   neighbor 100.100.100.100 pw-id 1
  p2p pwhe2
   interface pw-ether2
   neighbor 100.100.100.100 pw-id 2
```

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 GigabitEthernet 0/0/0/0.1
        neighbor 1.1.1.1 pw-id 1
            pw-class class1
        !
    !
!
```

Enabling Pseudowire Grouping: Example

This example shows how to enable pseudowire grouping.

```
config
l2vpn
  pw-grouping
```

Additional References

For additional information related to implementing VPLS, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS VPLS-related commands	*MPLS Virtual Private LAN Services Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*

Related Topic	Document Title
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
Cisco CRS router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

[1] Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 4447	*Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)*, April 2006
RFC 4448	*Encapsulation Methods for Transport of Ethernet over MPLS Networks*, April 2006

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing IPv6 VPN Provider Edge Transport over MPLS

IPv6 VPN Provider Edge (6PE) uses the existing MPLS IPv4 core infrastructure for IPv6 transport. 6PE enables IPv6 sites to communicate with each other over an MPLS IPv4 core network using MPLS label switched paths (LSPs).

This feature relies heavily on multiprotocol Border Gateway Protocol (BGP) extensions in the IPv4 network configuration on the provider edge (PE) router to exchange IPv6 reachability information (in addition to an MPLS label) for each IPv6 address prefix. Edge routers are configured as dual-stack, running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange.

For detailed information about the commands used to configure L2TP functionality, see *Cisco IOS XR Routing Command Reference*.

Feature History for Implementing 6PE on Cisco IOS XR Software

Release	Modification
Release 3.5.0	This feature was introduced.
Release 3.7.0	Support was added for: • Inter-AS 6PE

Contents

Prerequisites for Implementing 6PE

The following prerequisites are required to implement 6PE:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

  If you need assistance with your task group assignment, contact your system administrator.
- You must be familiar with MPLS and BGP4 configuration and troubleshooting.

Information About 6PE

To configure the 6PE feature, you should understand the following concepts, which are described in the following sections:

- Overview of 6PE, page VPC-188
- Benefits of 6PE, page VPC-188
- Deploying IPv6 over MPLS Backbones, page VPC-189
- IPv6 on the Provider Edge and Customer Edge Routers, page VPC-189
- IPv6 Provider Edge Multipath, page VPC-190

Overview of 6PE

Multiple techniques are available to integrate IPv6 services over service provider core backbones:

- Dedicated IPv6 network running over various data link layers
- Dual-stack IPv4-IPv6 backbone
- Leveraging of an existing MPLS backbone

These solutions are deployed on service providers' backbones when the amount of IPv6 traffic and the revenue generated are in line with the necessary investments and the risks agreed to. Conditions are favorable for the introduction of native IPv6 service, from the edge, in a scalable way, without any IPv6 addressing restrictions and without putting a well-controlled IPv4 backbone in jeopardy. Backbone stability is key for service providers that recently stabilized their IPv4 infrastructure.

Service providers running an MPLS/IPv4 infrastructure follow the same trends, as several integration scenarios are possible to offer IPv6 services on an MPLS network. Cisco Systems specially developed Cisco 6PE, or, IPv6 Provider Edge Router over MPLS, to meet all of those requirements.

Inter-AS support for 6PE requires support of Border Gateway Protocol (BGP) to enable the address families and to allocate and distribute the PE and ASBR labels.

Benefits of 6PE

Service providers that currently deploy MPLS will experience the following benefits of Cisco 6PE:

- Minimal operational cost and risk—No impact on existing IPv4 and MPLS services.

- Provider edge routers upgrade only—A 6PE router can be an existing PE router or a new one dedicated to IPv6 traffic.
- No impact on IPv6 customer edge routers—The ISP can connect to any customer CE running Static, IGP or EGP.
- Ready for production services—An ISP can delegate IPv6 prefixes.
- IPv6 introduction into an existing MPLS service—6PE routers can be added at any time.
- It is possible to switch up to OC-192 speed in the core.

Deploying IPv6 over MPLS Backbones

Backbones enabled by 6PE (IPv6 over MPLS) allow IPv6 domains to communicate with each other over an MPLS IPv4 core network. This implementation requires no backbone infrastructure upgrades and no reconfiguration of core routers, because forwarding is based on labels rather than on the IP header itself. This provides a very cost-effective strategy for IPv6 deployment.

Additionally, the inherent virtual private network (VPN) and traffic engineering (TE) services available within an MPLS environment allow IPv6 networks to be combined into VPNs or extranets over an infrastructure that supports IPv4 VPNs and MPLS-TE.

IPv6 on the Provider Edge and Customer Edge Routers

Service Provider Edge Routers

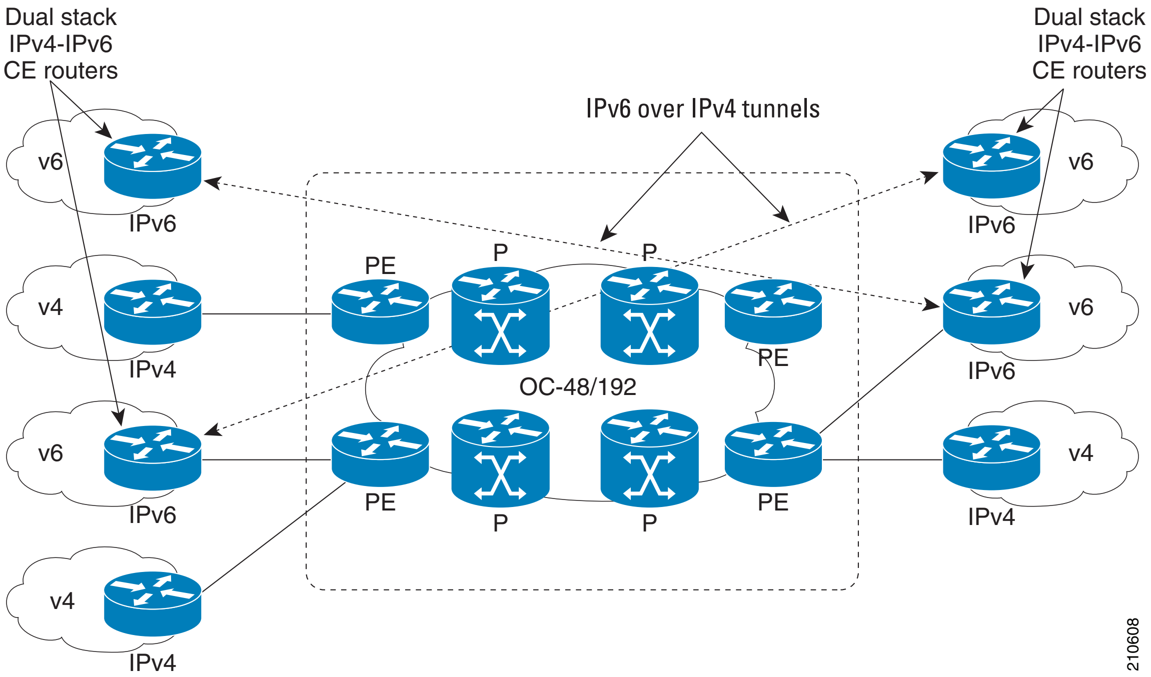
6PE is particularly applicable to service providers who currently run an MPLS network. One of its advantages is that there is no need to upgrade the hardware, software, or configuration of the core network, and it eliminates the impact on the operations and the revenues generated by the existing IPv4 traffic. MPLS is used by many service providers to deliver services to customers. MPLS as a multiservice infrastructure technology is able to provide layer 3 VPN, QoS, traffic engineering, fast re-routing and integration of ATM and IP switching.

Customer Edge Routers

Using tunnels on the CE routers is the simplest way to deploy IPv6 over MPLS networks. It has no impact on the operation or infrastructure of MPLS and requires no changes to the P routers in the core or to the PE routers. However, tunnel meshing is required as the number of CEs to connect increases, and it is difficult to delegate a global IPv6 prefix for an ISP.

Figure 21 illustrates the network architecture using tunnels on the CE routers.

Figure 21 IPv6 Using Tunnels on the CE Routers



IPv6 Provider Edge Multipath

Internal and external BGP multipath for IPv6 allows the IPv6 router to load balance between several paths (for example, same neighboring autonomous system (AS) or sub-AS, or the same metric) to reach its destination. The 6PE multipath feature uses multiprotocol internal BGP (MP-IBGP) to distribute IPv6 routes over the MPLS IPv4 core network and to attach an MPLS label to each route.

When MP-IBGP multipath is enabled on the 6PE router, all labeled paths are installed in the forwarding table with MPLS information (label stack) when MPLS information is available. This functionality enables 6PE to perform load balancing.

How to Implement 6PE

This section includes the following implementation procedure:

- Configuring 6PE, page VPC-190

Configuring 6PE

This task describes how to configure 6PE on PE routers to transport the IPv6 prefixes across the IPv4 cloud.

Be sure to configure 6PE on PE routers participating in both the IPv4 cloud and IPv6 clouds.

Note To learn routes from both clouds, you can use all routing protocols supported on Cisco IOS XR software: BGP, OSPF, IS-IS, EIGRP, RIP, and Static.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **address-family ipv6 labeled-unicast**
5. **exit**
6. **exit**
7. **address-family ipv6 unicast**
8. **allocate-label** [**all** | **route-policy** *policy_name*]
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `RP/0/0/CPU0:router(config)# router bgp 1`	Enters the number that identifies the autonomous system (AS) in which the router resides. Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/0/CPU0:router(config-bgp)# neighbor 1.1.1.1`	Enters neighbor configuration mode for configuring Border Gateway Protocol (BGP) routing sessions.
Step 4	**address-family ipv6 labeled-unicast** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv6 labeled-unicast`	Specifies IPv6 labeled-unicast address prefixes. **Note** This option is also available in IPv6 neighbor configuration mode and VRF neighbor configuration mode.
Step 5	**exit** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# exit`	Exits BGP address-family submode.

	Command or Action	Purpose

Step 6	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# exit`
		Exits BGP neighbor submode.

Step 7	**`address-family ipv6 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family ipv6 unicast`
		Specifies IPv6 unicast address prefixes.

Step 8	**`allocate-label`** `[`**`all`** `|` **`route-policy`** *`policy_name`*`]`
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# allocate-label all`
		Allocates MPLS labels for specified IPv4 unicast routes.
		Note The **route-policy** keyword provides finer control to filter out certain routes from being advertised to the neighbor.

Step 9	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-af)# 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.

Configuration Examples for 6PE

This section includes the following configuration example:

- Configuring 6PE on a PE Router: Example, page VPC-192

Configuring 6PE on a PE Router: Example

The following sample configuration shows the configuration of 6PE on a PE router:

```
interface GigabitEthernet0/3/0/0
 ipv6 address 2001::1/64
!
```

```
router isis ipv6-cloud
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface GigabitEthernet0/3/0/0
  address-family ipv6 unicast
  !
!
router bgp 55400
 bgp router-id 54.6.1.1
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
  network 55:5::/64
  redistribute connected
  redistribute isis ipv6-cloud
 !
 neighbor 34.4.3.3
  remote-as 55400
  address-family ipv4 unicast
  !
  address-family ipv6 labeled-unicast
```

Additional References

For additional information related to this feature, refer to the following references:

Related Document

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco CRS router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

[1] Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
—	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing Layer 2 Tunnel Protocol Version 3

Layer 2 Tunnel Protocol Version 3 (L2TPv3) is an Internet Engineering Task Force (IETF) working group draft that provides several enhancements to L2TP, including the ability to tunnel any Layer 2 (L2) payload over L2TP. Specifically, L2TPv3 defines the L2TP protocol for tunneling Layer 2 payloads over an IP core network using L2 virtual private networks (VPNs).

For additional information about L2TPv3, see *MPLS VPNs over IP Tunnels on Cisco IOS XR Software.*

Feature History for Implementing Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR

Release	Modification
Release 3.7.0	This feature was introduced.
Release 3.8.0	Support was added for the following features on the Cisco XR 12000 Series Router: • IP Interworking on Engine 3 and 5 Line Cards • PPP/HDLC Like-to-Like Pseudowires on Engine 3 and Engine 5 Line Cards • Frame Relay DLCI, and MLFR Like-to-Like Pseudowires on Engine 3 Line Cards • Ethernet Port Mode and VLAN Like-to-Like on Engine 3 Line Cards • Local Switching Support with L2TPv3 on Engine 3 and 5 Line Cards
Release 4.0	Support was added for QinQ or QinAny Attachment Circuits over L2TPv3 core on the Engine 5 line cards. Support was added for the following features to implement L2TPv3 over 4 Port Channelized OC12 Engine 3 line cards: • IP Interworking (Frame Relay DLCI-to-ATM, Frame Relay DLCI-to-Ethernet (VLAN) and Frame Relay DLCI-to-Ethernet Port) • Frame Relay PVC DLCI Like-to-Like Pseudowires • PPP/HDLC Like-to-Like Pseudowires

Contents

Prerequisites for Layer 2 Tunnel Protocol Version 3

The following prerequisites are required to implement L2TPv3:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

  If you need assistance with your task group assignment, contact your system administrator.
- You must enable Cisco Express Forwarding (CEF) before you configure a cross-connect attachment circuit (AC) for a customer edge (CE) device.
- You must configure a Loopback interface on the router for originating and terminating the L2TPv3 traffic. The Loopback interface must have an IP address that is reachable from the remote provider edge (PE) device at the other end of an L2TPv3 control-channel.
- You must enable Simple Network Management Protocol (SNMP) notifications of L2TP session up and session down events.

Note A cross-connection is expressed as *xconnect* in the CLI.

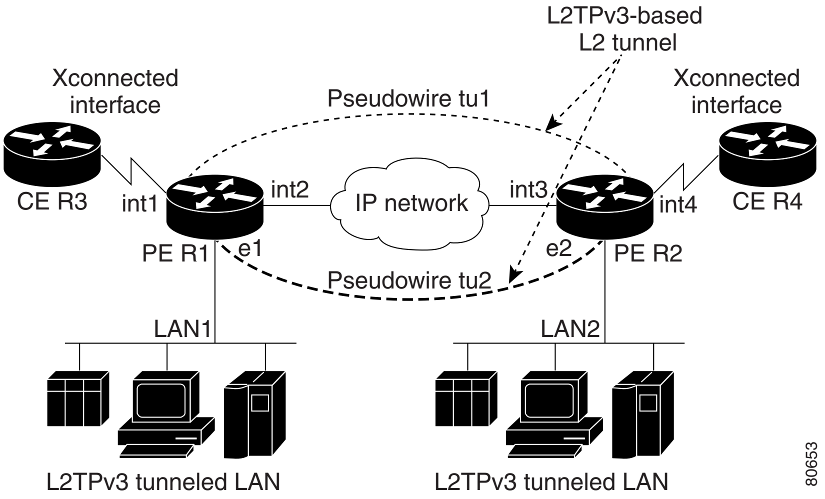
Information About Layer 2 Tunnel Protocol Version 3

To configure the L2TPv3 feature, you should understand the following concepts:

L2TPv3 Operation

Figure 22 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 22 L2TPv3 Operation



In Figure 22, 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.

L2TPv3 Benefits

L2TPv3 provides the following benefits:

- Simplifies deployment of VPNs—L2TPv3 is an industry-standard L2 tunneling protocol that ensures interoperability among vendors, increasing customer flexibility and service availability.
- Does not require MPLS—Service providers need not deploy MPLS in the core IP backbone to set up VPNs using L2TPv3 over the IP backbone; this will result in operational savings and increased revenue.
- Supports L2 tunneling over IP for any payload—L2TPv3 provides enhancements to L2TP to support L2 tunneling of any payload over an IP core network. L2TPv3 defines the base L2TP protocol as being separate from the L2 payload that is tunneled.

L2TPv3 Features

L2TPv3 provides cross-connect support for Ethernet, 802.1q (VLAN), Frame Relay, HDLC, and PPP, and ATM using the sessions described in the following sections:

- Static L2TPv3 Sessions, page VPC-198
- Dynamic L2TPv3 Sessions, page VPC-198

L2TPv3 also supports:

- Sequencing, page VPC-199

- Local Switching, page VPC-199
- Local Switching: Quality of Service, page VPC-201
- L2TPv3 Pseudowire Switching, page VPC-201
- L2TPv3 Pseudowire Manager, page VPC-201
- IP Packet Fragmentation, page VPC-201
- L2TPv3 Type of Service Marking, page VPC-202
- Keepalive, page VPC-202
- Maximum Transmission Unit Handling, page VPC-202
- Distributed switching
- L2TPv3 L2 fragmentation
- L2TPv3 control message hashing
- L2TPv3 control message rate limiting
- L2TPv3 digest secret graceful switchover
- Manual clearing of L2TPv3 tunnels
- L2TPv3 tunnel management
- Color aware policer on ethernet over L2TPv3
- Site of origin for BGP VPNs
- L2TPV3 IP Interworking, page VPC-202
- Like-to-Like Pseudowires, page VPC-203

Static L2TPv3 Sessions

Typically, the L2TP control plane is responsible for negotiating session parameters (such as the session ID or the cookie) to set up the session; however, some IP networks require sessions to be configured so that no signaling is required for session establishment. Therefore, you can set up static L2TPv3 sessions for a PE router by configuring fixed values for the fields in the L2TP data header. A static L2TPv3 session allows the PE to tunnel L2 traffic as soon as the AC to which the session is bound comes up.

Note In an L2TPv3 static session, you can still run the L2TP control-channel to perform peer authentication and dead-peer detection. If the L2TP control-channel cannot be established or is torn down because of a hello failure, the static session is also torn down.

When you use a static L2TPv3 session, you cannot perform circuit interworking (for example, LMI) because there is no facility to exchange control messages. To perform circuit interworking, you must use a dynamic session.

Dynamic L2TPv3 Sessions

A dynamic L2TP session is established through the exchange of control messages containing attribute-value pair (AVP). Each AVP contains information about the nature of the L2 link being forwarded: including the payload type, virtual circuit (VC) ID, and so on.

Multiple L2TP sessions can exist between a pair of PEs, and can be maintained by a single control-channel. Session IDs and cookies are dynamically generated and exchanged as part of a dynamic session setup. Sequencing configuration is also exchanged and circuit state changes are conveyed using the set link info (SLI) message.

Sequencing

Although the correct sequence of received L2 frames is guaranteed by some L2 technologies (by the nature of the link, such as a serial line) or the protocol itself, forwarded L2 frames may be lost, duplicated, or reordered when they traverse a network as IP packets. If the L2 protocol does not provide an explicit sequencing mechanism, you can configure L2TP to sequence its data packets according to the data channel sequencing mechanism described in the L2TPv3 IETF l2tpext working group draft.

A receiver of L2TP data packets mandates sequencing through the sequencing required AVP when the session is being negotiated. A sender that receives this AVP (or that is manually configured to send sequenced packets) uses the L2-specific pseudowire control encapsulation defined in L2TPv3.

Currently, you can configure L2TP only to drop out-of-order packets; you cannot configure L2TP to deliver the packets out-of-order. No reordering mechanism is available.

Local Switching

An AC to AC cross-connect, also called *local switching*, is a building block of L2VPN that allows frames to switch between two different ACs on the same PE. PE (see Figure 23).

You must configure separate IP addresses for each cross-connect statement on the Carrier Edge router.

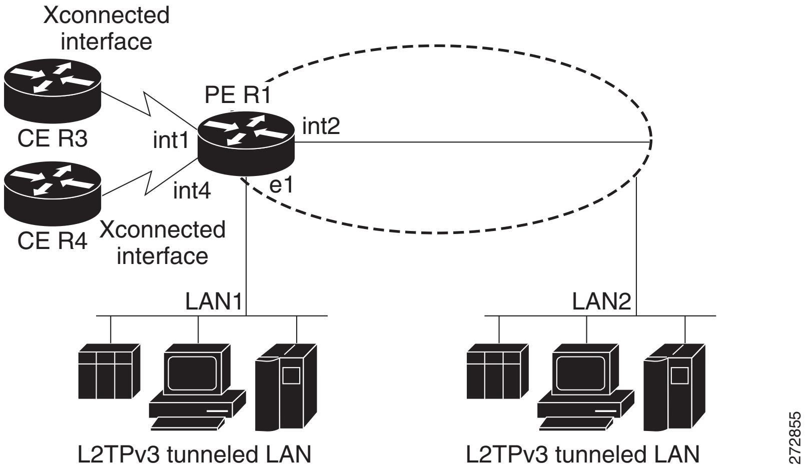
The following configurations are supported for local switching:

- IP interworking for Ethernet, Frame Relay and ATM.
- Like-to-like Pseudowires for point-to-point connections, High-Level Data Link Control (HDLC), and Ethernet, and Frame Relay.
- VLAN-to-VLAN
- Port-to-VLAN
- VLAN-to-Port
- Encapsulation-to-other end encapsulation
- Port-to-Port
- Dot1q-to-Dot1q
- QinQ-to-QinQ
- QinAny-to-QinAny
- Dot1q-to-QinQ
- QinQ-to-Dot1q
- QinQ-to-QinAny
- QinAny-to-QinQ

Note VLAN-to-VLAN options do not require interworking. Port-to-VLAN and VLAN-to-port do, and it is locally managed by the L2VPN application. If both interfaces are Ethernet VLAN, each reside on a single physical interface. By definition, local switching is not a pseudowire technology, because signaling protocols (such as LDP or L2TPv3) are not involved.

Note In Release 4.0, the QinQ or QinAny over L2TPv3 feature is supported only on the Engine 5 V2 SPAs. The Engine 3 line cards do not support this feature.

Figure 23 Local Switching Operation



Local Switching: Quality of Service

The following quality of service (QoS) requirements apply to local switching:

- QoS service policies can be applied to any L2 AC (port or VLAN, or both) and can be applied to any interworking mode (port-to-port, vlan-to-port, port-to-vlan, vlan-to-vlan). The AC can be cross-connected to a pseudowire (EoL2TPv3) or to another AC (local switching).
- QoS service policies can be attached directly to the AC.
- QoS service policies can be attached to the main interface using **match vlan** on L2 VLAN ACs.
- QoS service policies attached to the main interface can be inherited by all L2 VLANs.
- QoS service policies cannot be attached to a main interface when there are service policies already attached to its L3VLANs or L2VLAN ACs.
- QoS service policies already attached to the main interface are not permitted on L3 VLAN or L2 VLAN ACs.

L2TPv3 Pseudowire Switching

L2VPN pseudowire switching allows you to:

- Extend L2VPN pseudowires across an Inter-AS boundary.
- Connect two or more contiguous pseudowire segments to form an end-to-end multihop pseudowire.
- Keep the IP addresses of the edge PE routers private across Inter-AS boundaries.
- Keep different administrative or provisioning domains to manage the end-to-end service.

L2TPv3 Pseudowire Manager

The pseudowire manager is a client library provided by the pseudowire signaling module that runs in the context of the L2VPN process. This client library implements interface to pseudo-wire signaling protocol for specific pseudowire type.

IP Packet Fragmentation

It is desirable to avoid fragmentation issues in the service provider network because reassembly is computationally expensive. The easiest way to avoid fragmentation issues is to configure the CE routers with an Maximum Transmission Unit (MTU) value that is smaller than the pseudowire path MTU. However, in scenarios where this is not an option, fragmentation issues must be considered. Previously, L2TP supported only the following options for packet fragmentation when a packet is determined to exceed the L2TP path MTU:

- Unconditionally drop the packet
- Fragment the packet after L2TP/IP encapsulation
- Drop the packet and send an Internet Control Message Protocol (ICMP) unreachable message back to the CE router

Currently, the following options for packet fragmentation are supported:

- Path MTU is a configurable value which is configured on PE. If the packet size and the L2TP header size are larger than the configured path MTU, packets are dropped.
- The PE configuration requires that a backbone facing interface's MTU is always greater or equal to the customer facing interface’s MTU and L2TP header size.

- IP fragmentation is not supported with L2TPv3.

L2TPv3 Type of Service Marking

When L2 traffic is tunneled across an IP network, information contained in the type of service (ToS) bits may be transferred to the L2TP-encapsulated IP packets in one of the following ways:

- If the tunneled L2 frames encapsulate IP packets themselves, it may be desirable to simply copy the ToS bytes of the inner IP packets to the outer IP packet headers. This action is known as “ToS byte reflection.”
- Static ToS byte configuration. You specify the ToS byte value used by all packets sent across the pseudowire.

Keepalive

The keepalive mechanism for L2TPv3 extends only to the endpoints of the tunneling protocol. L2TP has a reliable control message delivery mechanism that serves as the basis for the keepalive mechanism. The keepalive mechanism consists of an exchange of L2TP hello messages.

If a keepalive mechanism is required, the control plane is used, although it may not be used to bring up sessions. You can manually configure sessions.

In the case of static L2TPv3 sessions, a control channel between the two L2TP peers is negotiated through the exchange of start control channel request (SCCRQ), start control channel replay (SCCRP), and start control channel connected (SCCCN) control messages. The control channel is responsible only for maintaining the keepalive mechanism through the exchange of hello messages.

The interval between hello messages is configurable per control channel. If one peer detects that the other has gone down through the keepalive mechanism, it sends a StopCCN control message and then notifies all of the pseudowires to the peer about the event. This notification results in the teardown of both manually configured and dynamic sessions.

Maximum Transmission Unit Handling

It is important that you configure an maximum transmission unit (MTU) appropriate for a each L2TPv3 tunneled link. The configured MTU size ensures that the lengths of the tunneled L2 frames fall below the MTU of the destination AC.

L2TPv3 handles the MTU as follows:

- Configure the path MTU on the PE. If the packet size and the L2TP header collectively are larger than the configured value, packets are dropped.

L2TPV3 IP Interworking

IP Interworking, also known as *routed interworking,* is a way in which diverse transports are interconnected to each other over a Layer 2 transport such as L2TPv3. For example, a Frame Relay DLCI could be connected at one end to an Ethernet VLAN at the other. This kind of interconnection is normal for Layer 3 connections where the Layer 2 encapsulation is disregarded and only the inner Layer 3 packet is transported over the network. IP Interworking performs the same function, except that it does not route based on the Layer 3 IP address. Instead, it uses a fixed point-to-point connection per session based on user configuration, and signaled by the L2TPv3 control plane.

The prerequisite to IP Interworking is that the payload being transported over a pseudowire is an IP payload. Non-IP packets are not transported over the pseudowire.

The following modes support interworking in L2TPv3 on the Engine-5 line cards:

- Ethernet Port Mode and VLAN mode, page VPC-203
- Frame Relay Point-to-Point DLCI and MLFR, page VPC-203
- ATM (AAL5), page VPC-203

Ethernet Port Mode and VLAN mode

In the Ethernet Port mode, the Ethernet header is removed during encapsulation and only the inner IP packet is encapsulated with L2TPv3 headers and sent across the pseudowire. Only non-broadcast mode is supported and only one MAC address is associated with a single VLAN. If the Q-in-Q mode is not supported, then those frames are dropped.

During decapsulation, the L2TPv3 headers are removed and the appropriate ethernet header is placed before the IP packet and this is transmitted to the customer edge router. A broadcast address is used until the correct MAC address is identified. The Provider Edge router sends Internet Router Discovery Protocol (IRDP) messages over the ethernet link to get the MAC address from the Customer Edge router. The CE must be configured to receive and respond to IRDP.

Frame Relay Point-to-Point DLCI and MLFR

In the Frame Relay DLCI mode of IP interworking, the Frame Relay header is removed during encapsulation and only the inner IP packet is encapsulated with L2TPv3 headers and sent across the pseudowire. During decapsulation, the L2TPv3 headers are removed and the Frame Relay header and DLCI are placed before the IP packet. This is transmitted to the customer edge router.

ATM (AAL5)

IP interworking for ATM in L2TPv3 is supported only in the ATM adaptation layer 5 (AAL5) mode as this mode supports IP packets as payload, and these packets can be extracted. In other modes such as cell relay modes, there is no standard to identify the IP payload.

For IP interworking in ATM, the ATM headers are removed during encapsulation and only the inner IP packet is encapsulated with L2TPv3 headers and transported across the pseudowire.

Note A Layer 2 header is not transported over the pseudowire from the remote end. It must be manually added during decapsulation. LMI or other control frames arealso not carried from the remote end, therefore these cannot be sent out as decapsulated packets.

Like-to-Like Pseudowires

A PseudoWire (PW) is a bidirectional virtual circuit (VC) connecting two Attached Circuits (ACs). In an MPLS network, PWs are carried inside an LSP tunnel.

The ATM like-to-like pseudowires support the following modes:

- Cell relay
- Cell packing
- ATM adaptation layer 5 (AAL5)

The following features describe the pseudowire connection:

- PPP/HDLC, page VPC-204
- Frame Relay DLCI and MLFR, page VPC-204

PPP/HDLC

A point-to-point (PPP) connection allows service providers to provide a transparent PPP pass-through where the customer-edge routers can exchange the traffic through an end-to-end PPP session. Service providers can offer a virtual leased-line solution, and use the PPP subinterface capability to peer with multiple providers through a single POS connection.

A High Level Data Link Control (HDLC) connection is emulated from a customer router to another customer router across an IPv4 backbone. This technology allows transportation of HDLC frames across the packet networks.

The HDLC pseudowire over a Layer 2 Tunnel Protocol is intended to operate in Port mode, passing all HDLC data and protocol data units (PDU) over the pseudowire. Since all packets are passed in a largely transparent manner over the pseuwire, any protocol that has HDLC-like framing may utilize the HDLC pseudowire mode. In such cases, the negotiations and signaling of the specific protocols transported occur between the Remote Systems.

Frame Relay DLCI and MLFR

Frame Relay DLCIs are connected to create an end-to-end Frame Relay permanent virtual circuit (PVC). Traffic arriving on a DLCI on one interface is forwarded across the pseudowire to another DLCI on the other interface. The carrier edge devices may be a Frame Relay switch or an end-user device. Each Frame Relay PVC is composed of multiple segments. The DLCI value is local to each segment and is changed as traffic is switched from segment to segment.

The Multilink Frame Relay (MLFR) functionality is based on the Frame Relay Forum Multilink Frame Relay UNI/NNI Implementation Agreement (FRF.16). This feature provides a cost-effective way to increase bandwidth for particular applications by enabling multiple serial links to be aggregated into a single bundle of bandwidth.

How to Implement Layer 2 Tunnel Protocol Version 3

This section includes the tasks required to implement L2TPv3, as follows:

- Configuring a Pseudowire Class, page VPC-205 (required)
- Configuring L2TP Control-Channel Parameters, page VPC-207 (required)
- Configuring L2TPv3 Pseudowires, page VPC-217 (required)
- Configuring the Cross-connect Attachment Circuit, page VPC-224 (required)
- Configuring L2TPv3 IP Interworking, page VPC-226 (required)

Configuring a Pseudowire Class

Perform this task to configure a pseudowire class, or template.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **pw-class** *class name*
4. **encapsulation** {**mpls** | **l2tpv3**}
5. **sequencing** {**both**}
6. **protocol l2tpv3 class** *class name*
7. **ipv4 source** *ip-address*
8. **transport mode** {**ethernet** | **vlan**}
9. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enter L2VPN configure submode.

Step 3	**`pw-class`** *`class name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# pw-class wkg`
		Enters a pseudowire-class name.

	Command or Action	Purpose
Step 4	**encapsulation l2tpv3** **Example:** `RP/0/0/CPU0:router(config-l2tp-pwc)# encapsulation l2tpv3`	Configures pseudowire encapsulation to the Layer 2 Tunnel Protocol.

Step 5	**sequencing** {**both**}
	Example:
	`RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# sequencing both`
		Configures pseudowire class sequencing.

Step 6	**protocol l2tpv3 class** *class name* **Example:** `RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# protocol l2tpv3 class Class_l2tp_01`	Configures the L2TPv3 dynamic pseudowire signaling protocol to be used to manage the pseudowires created. **Note** Ensure that the L2TPv3 class name begins with a letter (A to Z or a to z). The class name can contain letters (A to Z or a to z) or numbers (0 to 9) and other characters such as underscore (_), hyphen (-) or period (.). A maximum of 31 characters can be used in the class name.
Step 7	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# ipv4 source 126.10.1.55`	Configures the local source IPv4 address.

Step 8	**transport-mode** {**ethernet** | **vlan**}
	Example:
	`RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# transport-mode ethernet`
		Configures the remote transport mode.

Step 9	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# end`
	or
	`RP/0/0/CPU0:router(config-l2tp-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 L2TP Control-Channel Parameters

This section describes the tasks you must perform to create a template of L2TP control-channel parameters that can be inherited by different pseudowire classes. The three main parameters described are:

- Timing parameters
- Authentication parameters
- Maintenance 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 three main groups of L2TP control-channel parameters that you can configure in an L2TP class are described in the following subsections:

- Configuring L2TP Control-Channel Timing Parameters, page VPC-207
- Configuring L2TPv3 Control-Channel Authentication Parameters, page VPC-208
- Configuring L2TP Control-Channel Maintenance Parameters, page VPC-217

Note When you enter L2TP class configuration mode, you can configure L2TP control-channel parameters in any order. If you have multiple authentication requirements you can configure multiple sets of L2TP class control-channel parameters with different L2TP class names. However, only one set of L2TP class control-channel parameters can be applied to a connection between any pair of IP addresses.

Configuring L2TP Control-Channel Timing Parameters

The following L2TP control-channel timing parameters can be configured in L2TP class configuration mode:

- Packet size of the receive window used for the control-channel.
- Retransmission parameters used for control messages.
- Timeout parameters used for the control-channel.

Note This task configures a set of timing control-channel parameters in an L2TP class. All timing control-channel parameter configurations can be configured in any order. If not configured, the default values are applied.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *l2tp-class-name*
3. **receive-window** *size*
4. **retransmit** *{***initial retries** *initial-retries* | **retries** *retries* | **timeout** *{***max** | **min***} timeout}*
5. **timeout setup** *seconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *l2tp-class-name* **Example:** `RP/0/0/CPU0:router(config)#` **l2tp-class cisco**	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 3	**receive-window** *size* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# receive-window 30`	Configures the number of packets that can be received by the remote peer before backoff queueing occurs. The default value is 512.

Step 4	**retransmit** {**initial retries** *initial-retries* | **retries** *retries* | **timeout** {**max** | **min**} *timeout*}
	Example:
	`RP/0/0/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 5	**timeout setup** *seconds* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# timeout setup 400`	Configures the amount of time, in seconds, allowed to set up a control-channel. • Range is 60 to 6000. Default value is 300.

Configuring L2TPv3 Control-Channel Authentication Parameters

Two methods of control-channel message authentication are available:

- L2TP Control-Channel (see Configuring Authentication for the L2TP Control-Channel, page VPC-209)
- L2TPv3 Control Message Hashing (see Configuring L2TPv3 Control Message Hashing, page VPC-210)

You can enable both methods of authentication to ensure interoperability with peers that support only one of these methods of authentication, but this configuration will yield control of which authentication method is used to the peer PE router. Enabling both methods of authentication should be considered an interim solution to solve backward-compatibility issues during software upgrades.

The principal difference between the L2TPv3 Control Message Hashing feature and CHAP-style L2TP control-channel authentication is that, instead of computing the hash over selected contents of a received control message, the L2TPv3 Control Message Hashing feature uses the entire message in the hash. In addition, instead of including the hash digest in only the SCCRP and SCCCN messages, it includes it in all messages.

This section also describes how to configure L2TPv3 digest secret graceful switchover (see Configuring L2TPv3 Digest Secret Graceful Switchover, page VPC-212,) which lets you make the transition from an old L2TPv3 control-channel authentication password to a new L2TPv3 control-channel authentication password without disrupting established L2TPv3 tunnels.

Note Support for L2TP control-channel authentication is maintained for backward compatibility. Either or both authentication methods can be enabled to allow interoperability with peers supporting only one of the authentication methods.

Configuring Authentication for the L2TP Control-Channel

The L2TP control-channel method of authentication is the older, CHAP-like authentication system inherited from L2TPv2.

The following L2TP control-channel authentication parameters can be configured in L2TP class configuration mode:

- Authentication for the L2TP control-channel
- Password used for L2TP control-channel authentication
- Local hostname used for authenticating the control-channel

This task configures a set of authentication control-channel parameters in an L2TP class. All of the authentication control-channel parameter configurations may be configured in any order. If these parameters are not configured, the default values are applied.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **authentication**
4. **password** {**0** | **7**} *password*
5. **hostname** *name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 3	**authentication** **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# authentication`	Enables authentication for the control-channel between PE routers.
Step 4	**password** {**0** | **7**} *password* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# password 7 cisco`	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	**hostname** *name* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# hostname yb2`	Specifies a hostname used to identify the router during L2TP control-channel authentication. • If you do not use this command, the default hostname of the router is used.

Configuring L2TPv3 Control Message Hashing

Perform this task to configure L2TPv3 Control Message Hashing feature for an L2TP class.

L2TPv3 control message hashing incorporates authentication or integrity check for all control messages. This per-message authentication is designed to guard against control message spoofing and replay attacks that would otherwise be trivial to mount against the network.

Enabling the L2TPv3Control Message Hashing feature will impact performance during control-channel and session establishment because additional digest calculation of the full message content is required for each sent and received control message. This is an expected trade-off for the additional security afforded by this feature. In addition, network congestion may occur if the receive window size is too small. If the L2TPv3 Control Message Hashing feature is enabled, message digest validation must be enabled. Message digest validation deactivates the data path received sequence number update and restricts the minimum local receive window size to 35.

You can configure control-channel authentication or control message integrity checking; however, control-channel authentication requires participation by both peers, and a shared secret must be configured on both routers. Control message integrity check is unidirectional, and requires configuration on only one of the peers.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*]
4. **hidden**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`configure`
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	`l2tp-class` *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.

	Command or Action	Purpose
Step 3	**digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*] **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# digest secret cisco hash sha`	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 4	**hidden** **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# hidden`	Enables AVP hiding when sending control messages to an L2TPv3 peer.

Configuring L2TPv3 Digest Secret Graceful Switchover

Perform this task to make the transition from an old L2TPv3 control-channel authentication password to a new L2TPv3 control-channel authentication password without disrupting established L2TPv3 tunnels.

Note This task is not compatible with authentication passwords configured with the older, CHAP-like control-channel authentication system.

L2TPv3 control-channel authentication occurs using a password that is configured on all participating peer PE routers. The L2TPv3 Digest Secret Graceful Switchover feature allows a transition from an old control-channel authentication password to a new control-channel authentication password without disrupting established L2TPv3 tunnels.

Before performing this task, you must enable control-channel authentication (see Configuring L2TPv3 Control Message Hashing, page VPC-210).

Note During the period when both a new and an old password are configured, authentication can occur only with the new password if the attempt to authenticate using the old password fails.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*]
4. **end**
or
commit
5. **show l2tp tunnel** brief
6. **configure**
7. **l2tp-class** *word*
8. **no digest** [**secret** [**0** | **7**] *password*] [**hash** {**md5** | **sha**}]
9. **end**
or
commit
10. **show l2tp tunnel** brief

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2tp-class`** *`word`*
	Example:
	`RP/0/0/CPU0:router(config)# l2tp-class class1`
		Specifies the L2TP class name and enters L2TP class configuration mode.

	Command or Action	Purpose

Step 3	**`digest`** `{`**`check disable`** `|` **`hash`** `{`**`MD5`** `|` **`SHA1`**`}] |` **`secret`** `{`**`0`** `|` **`7`**`}` *`password`*`]`
	Example:
	`RP/0/0/CPU0:router(config-l2tp-class)# digest secret cisco hash sha`
		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 4	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2tp-class)# end`
	or
	`RP/0/0/CPU0:router(config-l2tp-class)# commit`
		Saves configuration changes.
		- When you enter 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.
		- When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

	Command or Action	Purpose
Step 5	**show l2tp tunnel brief** **Example:** `RP/0/0/CPU0:router# show l2tun tunnel brief`	Displays the current state of L2 tunnels and information about configured tunnels, including local and remote L2 Tunneling Protocol (L2TP) hostnames, aggregate packet counts, and control-channel information. **Note** Use this command to determine if any tunnels are not using the new password for control-channel authentication. The output displayed for each tunnel in the specified L2TP class should show that two secrets are configured.
Step 6	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 7	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 8	**no digest** {**check disable** | **hash** {**MD5** | **SHA1**}] | **secret** {**0** | **7**} *password*] **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# no digest secret cisco hash sha1`	Disables L2TPv3 control-channel authentication or integrity checking.

	Command or Action	Purpose

Step 9	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2tp-class)# end`
	or
	`RP/0/0/CPU0:router(config-l2tp-class)# 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 10	**show l2tp tunnel brief** **Example:** `RP/0/0/CPU0:router# show l2tun tunnel brief`	Displays the current state of L2 tunnels and information about configured tunnels, including local and remote L2 Tunneling Protocol (L2TP) hostnames, aggregate packet counts, and control-channel information. Tunnels should no longer be using the old control-channel authentication password. If a tunnel does not update to show that only one secret is configured after several minutes have passed, that tunnel can be manually cleared and a defect report should be filed with TAC. **Note** Issue this command to ensure that all tunnels are using only the new password for control-channel authentication. The output displayed for each tunnel in the specified L2TP class should show that one secret is configured.

Configuring L2TP Control-Channel Maintenance Parameters

Perform this task to configure the interval used for hello messages in an L2TP class.

SUMMARY STEPS

1. **configure**
2. **l2tp-class** *word*
3. **hello** *interval*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2tp-class** *word* **Example:** `RP/0/0/CPU0:router(config)# l2tp-class class1`	Specifies the L2TP class name and enters L2TP class configuration mode.
Step 3	**hello** *interval* **Example:** `RP/0/0/CPU0:router(config-l2tp-class)# hello 100`	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.

Configuring L2TPv3 Pseudowires

Perform the following tasks to configure static and dynamic L2TPv3 pseudowires:

- Configuring a Dynamic L2TPv3 Pseudowire, page VPC-217
- Configuring aStatic L2TPv3 Pseudowire, page VPC-220

Configuring a Dynamic L2TPv3 Pseudowire

Perform this task to configure a dynamic L2TPv3 pseudowire.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *name*
4. **p2p** *name*
5. **neighbor** *ip-address* **pw-id** *number*

6. **pw-class** *pw-class-name*
7. **end**
or
commit
8. **pw-class** *pw-class-name*
9. **encapsulation l2tpv3**
10. **protocol l2tpv3 class** *class-name*
11. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enter L2VPN configure submode.

Step 3	**`xconnect group`** *`name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group grp_01`
		Enter a name for the cross-connect group.

Step 4	**`p2p`** *`name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1`
		Enters p2p configuration submode to configure point-to-point cross-connects.

Step 5	**`neighbor`** *`ip-address`* **`pw-id`** *`number`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 665`
		Configures a pseudowire for a cross-connect.

Step 6	**`pw-class`** *`pw-class-name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class atom`
		Enters pseudowire class submode to define a name for the cross-connect.

	Command or Action	Purpose

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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.

Step 8	**pw-class** *pw-class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class class100`	Enters pseudowire class submode to define a pseudowire class template.

Step 9	`encapsulation l2tpv3`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`
		Configures L2TPv3 pseudowire encapsulation.

	Command or Action	Purpose
Step 10	**protocol l2tpv3 class** *class name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# protocol l2tpv3 class wkg`	Configures the dynamic pseudowire signaling protocol.

Step 11	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# commit`
		Saves configuration changes.
		• When you enter 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.
		• When you enter the **commit** command, the system saves the configuration changes to the running configuration file and remains within the configuration session.

Configuring aStatic L2TPv3 Pseudowire

Perform this task to configure a static L2TPv3 pseudowire.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *name*
4. **p2p** *name*
5. **neighbor** *ip-address* **pw-id** *number*
6. **l2tp static local session** {*session-id*}
7. **l2tp static local cookie size** {**0** | **4** | **8**} [**value** {*low-value*} [{*high-value*}]]
8. **l2tp static remote session** {*session-id*}
9. **l2tp static remote cookie size** {**0** | **4** | **8**} [**value** {*low-value*} [{*high-value*}]]
10. **pw-class** *name*
11. **end**
    or
    commit
12. **configure**

13. **l2vpn**
14. **pw-class** *name*
15. **encapsulation l2tpv3**
16. **ipv4 source** *ip-address*
17. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/0/CPU0:router# configure	Enters global configuration mode.
Step 2	**l2vpn** **Example:** RP/0/0/CPU0:router(config)# l2vpn	Enter L2VPN configure submode.
Step 3	**xconnect group** *name* **Example:** RP/0/0/CPU0:router(config-l2vpn)# xconnect group customer_X	Enter a name for the cross-connect group.
Step 4	**p2p** *name* **Example:** RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1	Enters p2p configuration submode to configure point-to-point cross-connects.
Step 5	**neighbor** *ip-address* **pw-id** *number* **Example:** RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 666	Configures a pseudowire for a cross-connect.
Step 6	**l2tp static local session** {*session-id*} **Example:** RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static local session 147	Configures a L2TP pseudowire static session ID.
Step 7	***l2tp static local cookie size*** *{***0** | **4** | **8***}* *[***value** *{low-value} [{high-value}]]* **Example:** RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static local cookie size 4 value 0XA	Configures a L2TP pseudowire static session cookie.

	Command or Action	Purpose

Step 8	**`l2tp static remote session`** `{`*`session-id`*`}`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static remote session 123`
		Configures a L2TP pseudowire remote session ID.

Step 9	**`l2tp static remote cookie size`** `{`*`0`*` | `*`4`*` | `*`8`*`} [`**`value`**` {`*`low-value`*`} [{`*`high-value`*`}]]`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# l2tp static remote cookie size 8 value 0x456 0xFFB`
		Configures a L2TP pseudowire remote session cookie.

Step 10	**`pw-class`** *`name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class atom`
		Enters pseudowire class submode to define a pseudowire class template.

Step 11	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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.

Step 12	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 13	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enter L2VPN configure submode.

	Command or Action	Purpose
Step 14	**pw-class** *name* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# pw-class class100`	Enters pseudowire class submode to define a pseudowire class template.
Step 15	**encapsulation l2tpv3** **Example:** `RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`	Configures L2TPv3 pseudowire encapsulation.
Step 16	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# ipv4 source 126.10.1.55`	Configures the local source IPv4 address.

Step 17	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# 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 the Cross-connect Attachment Circuit

This configuration procedure binds an Ethernet 802.1q VLAN, or Frame Relay AC to an L2TPv3 pseudowire for cross-connect service. The virtual circuit identifier that you configure creates the binding between a pseudowire configured on a PE router and an AC in a CE device. The virtual circuit identifier configured on the PE router at one end of the L2TPv3 control-channel must also be configured on the peer PE router at the other end.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *free_format_string*
4. **p2p** *name*
5. **interface** *interface_name*
6. **neighbor** *ip-address* **pw-id** *number*
7. **pw-class** *name*
8. **protocol l2tpv3 class** *class name*
9. **ipv4 sourc**e *ip-address*
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**l2vpn** **Example:** `RP/0/0/CPU0:router(config)# l2vpn`	Enter L2VPN configure submode.
Step 3	**xconnect group** *free_format_string* **Example:** `RP/0/0/CPU0:router(config-l2vpn)# xconnect group customer_X`	Configures a cross-connect group.
Step 4	**p2p** *xconnect_id* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1`	Enters p2p configuration submode to configure point-to-point cross-connects.

	Command or Action	Purpose
Step 5	**interface** *interface_name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/1/5/1`	Enters interface configuration mode.
Step 6	**neighbor** *ip-address* **pw-id** *number* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 666`	Configures a pseudowire for a cross-connect.
Step 7	**pw-class** *pw-class-name* **Example:** `RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class l2tpv3-encap`	Enters pseudowire class submode to define a pseudowire class template.
Step 8	**protocol l2tpv3 class** *class name* **Example:** `RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# protocol l2tpv3 class wkg`	Configures the dynamic pseudowire signaling protocol.
Step 9	**ipv4 source** *ip-address* **Example:** `RP/0/0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# ipv4 source 126.10.1.55`	Configures the local source IPv4 address.

Step 10	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p-pw)# end`
	or
	`RP/0/0/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 L2TPv3 IP Interworking

Perform these tasks to configure L2TPv3 IP routed Interworking.

SUMMARY STEPS

1. **configure**
2. **l2vpn**
3. **xconnect group** *free_format_string*
4. **p2p** *xconnect-id*
5. **interface** *type interface-path-id*
6. **pseudowire-class** *class name*
7. **encapsulation l2tpv3**
8. **interworking ipv4**
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`l2vpn`**
	Example:
	`RP/0/0/CPU0:router(config)# l2vpn`
		Enter L2VPN configure submode.

Step 3	**`xconnect group`** *`free_format_string`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# xconnect group customer_X`
		Configures a cross-connect group.

Step 4	**`p2p`** *`xconnect_id`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1`
		Enters p2p configuration submode to configure point-to-point cross-connects.

Step 5	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet 0/1/5/1`
		Enters interface configuration mode.

	Command or Action	Purpose

Step 6	**`pseudowire-class`** *`class name`*
	Example:
	`RP/0/0/CPU0:router(config-l2vpn)# pw-class X`
		Enters pseudowire class submode to define a pseudowire class template.

Step 7	**`encapsulation {l2tpv3}`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3`
		Specifies the tunneling encapsulation.

Step 8	**`interworking ipv4`**
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# interworking ip`
		Configures interworking on an IP v4 network.

Step 9	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3-interworking)# end`
	or
	`RP/0/0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3-interworking)# 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.

Configuration Examples for Layer 2 Tunnel Protocol Version 3

This section provides the following configuration examples:

Configuring an L2TP Class for L2TPv3-based L2VPN PE Routers: Example

The following example shows how to configure a L2TP class with L2TPv3 based L2VPN for a PE router.

```
configure
  l2tp-class Class_l2tp_01
    receive-window 256
    retransmit retries 8
    retransmit initial retries 10
    retransmit initial timeout max 4
    retransmit initial timeout min 2
    timeout setup 90
    hostname PE1
    hello-interval 100
    digest secret cisco hash MD5
  end
```

Configuring a Pseudowire Class: Example

The following example shows a pseudowire class configuration on a PE router:

```
configure
 l2vpn
  pw-class FR1
   encapsulation l2tpv3
    protocol l2tpv3 [class {class name}]}
    sequencing both [resync {5-65535}]
    dfbit set
    tos {reflect|value {value}}
    ttl {1-255}
    pmtu max {68-65535}
    ipv4 source {ipv4_address}
    cookie size {0|4|8}
```

Configuring L2TPv3 Control Channel Parameters: Example

The following example shows a typical L2TPv3 control-channel configuration:

```
configure
 l2tp-class FR-l2tp
  authentication
  hostname R2-PE1
  password 7 121A0C041104
  hello-interval 10
  digest secret 7 02050D480809
```

Configuring the Cross-Connect Group: Example

The following example shows how to group all cross -connects for FR1:

```
configure
l2vpn
 xconnect group PP-2101
  p2p xc2101
   interface GigabitEthernet0/4/0/5
   neighbor 150.150.150.250 pw-id 5
    pw-class l2tpv3_class100
   !
  !
```

Configuring an Interface for Layer 2 Transport Mode: Example

The following example shows how to configure an interface to operate in Layer 2 transport mode:

```
configure
 interface GigabitEthernet0/4/0/5 l2transport
  negotiation auto

l2vpn
 xconnect group PP-2101
  p2p xc2101
   interface GigabitEthernet0/4/0/5
   neighbor 150.150.150.250 pw-id 5
    pw-class l2tpv3_class100
   !
  !
```

Additional References

The following sections provide additional information related to L2TPv3.

Related Documents

Related Topic	Document Title
MPLS VPN-related commands	*MPLS Virtual Private Network Commands on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
MPLS Layer 2 VPNs	*Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
MPLS Layer 3 VPNs	*Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*

Related Topic	Document Title
MPLS VPNs over IP Tunnels	*MPLS VPNs over IP Tunnels on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Configuration Guide*
Cisco CRS router getting started material	*Cisco IOS XR Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
draft-ietf-l2tpext-l2tp-base-03.txt	Layer Two Tunneling Protocol (Version 3)'L2TPv3'

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1321	*The MD5 Message Digest Algorithm*
RFC 2104	*HMAC-Keyed Hashing for Message Authentication*
RFC 2661	*Layer Two Tunneling Protocol “L2TP”*
RFC 3931	*Layer Two Tunneling Protocol Version 3 “L2TPv3*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS VPNs over IP Tunnels

The MPLS VPNs over IP Tunnels feature lets you deploy Layer 3 Virtual Private Network (L3VPN) services, over an IP core network, using L2TPv3 multipoint tunneling instead of MPLS. This allows L2TPv3 tunnels to be configured as multipoint tunnels to transport IP VPN services across the core IP network.

Feature History for Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR

Release	Modification
Release 3.5.0	This feature was introduced.
Release 3.8.0	The Multiple Tunnel Source Address feature was supported.

Contents

Prerequisites for Configuring MPLS VPNs over IP Tunnels

The following prerequisites are required to implement MPLS VPNs over IP Tunnels:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

  If you need assistance with your task group assignment, contact your system administrator.
- You must be in a user group associated with a task group that includes the proper task IDs for
  - BGP commands
  - MPLS commands (generally)
  - MPLS Layer 3 VPN commands

Restrictions for Configuring MPLS VPNs over IP Tunnels

The following restriction applies when you configure MPLS VPNs over IP tunnels:

- MPLS forwarding cannot be enabled on a provider edge (PE) router.

Information About MPLS VPNs over IP Tunnels

To implement MPLS VPNs over IP Tunnels, you must understand the following concepts:

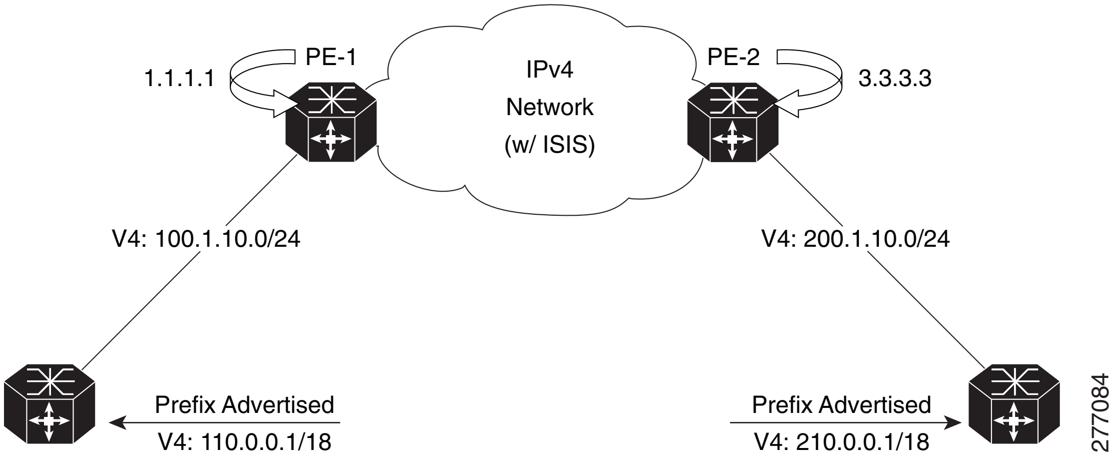
Overview: MPLS VPNs over IP Tunnels

Traditionally, VPN services are deployed over IP core networks using MPLS, *or* L2TPv3 tunnels using point-to-point links. However, an L2TPv3 multipoint tunnel network allows L3VPN services to be carried through the core without the configuration of MPLS.

L2TPv3 multipoint tunneling supports multiple tunnel endpoints, which creates a full-mesh topology that requires only one tunnel to be configured on each PE router. This permits VPN traffic to be carried from enterprise networks across cooperating service provider core networks to remote sites.

Figure 24 illustrates the topology used for the configuration steps.

Figure 24 Basic MPLS VPN over IP Topology



Advertising Tunnel Type and Tunnel Capabilities Between PE Routers—BGP

Border Gateway Protocol (BGP) is used to advertise the tunnel endpoints and the subaddress family identifier (SAFI) specific attributes (which contains the tunnel type, and tunnel capabilities). This feature introduces the tunnel SAFI and the BGP SAFI-Specific Attribute (SSA) attribute.

These attributes allow BGP to distribute tunnel encapsulation information between PE routers. VPNv4 traffic is routed through these tunnels. The next hop, advertised in BGP VPNv4 updates, determines which tunnel to use for routing tunnel traffic.

SAFI

The tunnel SAFI defines the tunnel endpoint and carries the endpoint IPv4 address and next hop. It is identified by the SAFI number 64.

BGP SSA

The BGP SSA carries the BGP preference and BGP flags. It also carries the tunnel cookie, tunnel cookie length, and session ID. It is identified by attribute number 19.

PE Routers and Address Space

One multipoint L2TPv3 tunnel must be configured on each PE router. To create the VPN, you must configure a unique Virtual Routing and Forwarding (VRF) instance. The tunnel that transports the VPN traffic across the core network resides in its own address space. A special purpose VRF called a *Resolve in VRF* (RiV) is created to manage the tunnel address space. You also configure the address space under the RiV that is associated with the tunnel and a static route in the RiV to route outgoing traffic through the tunnel.

Packet Validation Mechanism

The MPLS VPNs over IP Tunnels feature provides a simple mechanism to validate received packets from appropriate peers. The multipoint L2TPv3 tunnel header is automatically configured with a 64-bit cookie and L2TPv3 session ID. This packet validation mechanism protects the VPN from illegitimate traffic sources. The cookie and session ID are not user-configurable, but they are visible in the packet as it is routed between the two tunnel endpoints. Note that this packet validation mechanism does not protect the VPN from hackers who are able to monitor legitimate traffic between PE routers.

Quality of Service Using the Modular QoS CLI

To configure the bandwidth on the encapsulation and decapsulation interfaces, use the modular QoS CLI (MQC).

Note This task is optional.

Use the MQC to configure the IP precedence or Differentiated Services Code Point (DSCP) value set in the IP carrier header during packet encapsulation. To set these values, enter a standalone **set** command or a **police** command using the keyword **tunnel**. In the input policy on the encapsulation interface, you can set the precedence or DSCP value in the IP payload header by using MQC commands without the keyword **tunnel**.

Note You must attach a QoS policy to the physical interface—*not* to the tunnel interface.

If Modified Deficit Round Robin (MDRR)/Weighted Random Early Detection (WRED) is configured for the encapsulation interface in the input direction, the final value of the precedence or DSCP field in the IP carrier header is used to determine the precedence class for which the MDRR/WRED policy is applied. On the decapsulation interface in the input direction, you can configure a QoS policy based on the precedence or DSCP value in the IP carrier header of the received packet. In this case, an MQC policy with a class to match on precedence or DSCP value will match the precedence or DSCP value in the received IP carrier header. Similarly, the precedence class for which the MDRR/WRED policy is applied on the decapsulation input direction is also determined by precedence or DSCP value in the IP carrier header.

BGP Multipath Load Sharing for MPLS VPNs over IP Tunnels

BGP Multipath Load Sharing for EBGP and IBGP lets you configure multipath load balancing with both external BGP and internal BGP paths in BGP networks that are configured to use MPLS VPNs. (When faced with multiple routes to the same destination, BGP chooses the best route for routing traffic toward the destination so that no individual router is overburdened.)

BGP Multipath Load Sharing is useful for multihomed autonomous systems and PE routers that import both EBGP and IBGP paths from multihomed and stub networks.

Inter-AS and CSC Support over IP Tunnels

The L3VPN Inter-AS feature provides a method of interconnecting VPNs between different VPN service providers. Inter-AS supports connecting different VPN service providers to provide native IP L3VPN services. For more information about Inter-AS, see Implementing MPLS VPNs over IP Tunnels.

Carrier Supporting Carrier (CSC) is implemented in circumstances in which one service provider needs to use the transport services provided by another service provider. The service provider that provides the transport is called the backbone carrier. The service provider, which uses the services provided by the backbone carrier, is called a customer carrier. Backbone carriers with CSC, bridge two or more customer carrier sites through an MPLS VPN/MPLS VPN over IP tunnels backbone.

Multiple Tunnel Source Address

Currently, L2TPv3 tunnel encapsulation transports the VPN traffic across the IP core network between PEs with a /32 loopback addresses of PEs, and ingress PE uses a single /32 loopback address as the source IP address of tunnel encapsulation. This results in an imbalance on the load. In order to achieve load balance in the core, the ingress PE sends the VPN traffic with the source IP address of a L2TPv3 tunnel header taken from the pool for a /28 IP address instead of a single /32 address. This is called the Multiple Tunnel Source Address.

To support the /28 IP address, a keyword **source-pool** is used as an optional configuration command for the tunnel template. This keyword is located in the source address configuration. The source address is published to remote PEs through the BGP's tunnel SAFI messages.

Once the optional source-pool address is configured, it is sent to the forwarding information base (FIB). FIB uses a load balancing algorithm to get one address from the pool, and uses that address to call the tunnel infra DLL API to construct the tunnel encapsulation string.

The Multiple Tunnel Source Address infrastructure uses two primary models:

- Tunnel MA, page VPC-238
- Tunnel EA, page VPC-238

Tunnel MA

The Tunnel MA tunnel is used for the tunnel-template configuration and communicating with the BGP. It supports the /28 IP address by performing these basic tasks:

- Verifies and applies the /28 address pool configuration
- Extends the tunnel information to include the new address pool
- Sends the address pool information to Tunnel EA through the data path control (DPC)

Note Sending the address pool information to BGP is not mandatory.

Tunnel EA

Tunnel EA sends the address pool information to FIBand also supports the /28 IP address by performing these basic tasks:

- Processes the address pool information in the DPC from tunnel MA
- Saves the address pool information in the tunnel IDB in EA
- Sends the source address pool information to FIB

How to Configure MPLS VPNs over IP Tunnels

The following procedures are required to configure MPLS VPN over IP:

- Configuring the Global VRF Definition, page VPC-239 (required)
- Configuring a Route-Policy Definition, page VPC-241 (required)
- Configuring a Static Route, page VPC-241 (required)
- Configuring an IPv4 Loopback Interface, page VPC-243 (required)
- Configuring a CFI VRF Interface, page VPC-245 (required)
- Configuring the Core Network, page VPC-246 (required)
- Configuring Inter-AS and CSC Support over IP Tunnels, page VPC-247
- Verifying MPLS VPN over IP, page VPC-254 (optional)
- Configuring Source Pool Address for MPLS VPNs over IP Tunnels, page VPC-255 (optional)

Note All procedures occur on the local PE (PE1). Corresponding procedures must be configured on the remote PE (PE2).

Configuring the Global VRF Definition

Perform this task to configure the global VRF definition.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-target** [*0-65535.0-65535*:*0-65535* | *as-number:nn* | *ip-address:nn*]
5. **export route-target** [*0-65535.0-65535*:*0-65535* | *as-number:nn* | *ip-address:nn*]
6. **exit**
7. **address-family ipv6 unicast**
8. **import route-target** [*0-65535.0-65535*:*0-65535* | *as-number:nn* | *ip-address:nn*]
9. **export route-target** [*0-65535.0-65535*:*0-65535* | *as-number:nn* | *ip-address:nn*]
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config)# vrf vrf-name`
		Specifies a name assigned to a VRF.

Step 3	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast`
		Specifies an IPv4 address-family address.

Step 4	**`import route-target [`***`0-65535.0-65535:0-65535`*` | `*`as-number:nn`*` | `*`ip-address:nn`*`]`
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# import route-target 500:99`
		Configures a VPN routing and forwarding (VRF) import route-target extended community.

	Command or Action	Purpose
Step 5	**export route-target [***0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn***]** **Example:** `RP/0/0/CPU0:router(config-vrf-af)# export route-target 700:44`	Configures a VPN routing and forwarding (VRF) export route-target extended community.
Step 6	**exit** **Example:** `RP/0/0/CPU0:router(config-vrf-af)# exit`	Exits interface configuration mode.
Step 7	**address-family ipv6 unicast** **Example:** `RP/0/0/CPU0:router(config-vrf)# address-family ipv6 unicast`	Specifies an IPv6 address-family address.

Step 8	**import route-target [***0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn***]**
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# import route-target 500:99`
		Configures a VPN routing and forwarding (VRF) import route-target extended community.

Step 9	**export route-target [***0-65535.0-65535:0-65535* | *as-number:nn* | *ip-address:nn***]** **Example:** `RP/0/0/CPU0:router(config-vrf-af)# import route-target 700:88`	Configures a VPN routing and forwarding (VRF) export route-target extended community.

Step 10	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# end`
	or
	`RP/0/0/CPU0:router(config-vrf-af)# 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 a Route-Policy Definition

Perform this task to configure a route-policy definition for CE-PE EBGP.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name* **pass**
3. e**nd policy**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**route-policy** *name* **pass** **Example:** `RP/0/0/CPU0:router(config)# route-policy ottawa_admin pass`	Defines and passes a route policy.

Step 3	**end policy**
	Example:
	`RP/0/0/CPU0:router(config-rpl)# end policy`
		End of route-policy definition.

Configuring a Static Route

Perform this task to add more than 4K static routes (Global/VRF).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **maximum path ipv4** *1-140000*
4. **maximum path ipv6** *1-140000*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router static`**
	Example:
	`RP/0/0/CPU0:router(config)# router static`
		Enters static route configuration subcommands.

Step 3	**`maximum path ipv4`** *`1-140000`*
	Example:
	`RP/0/0/CPU0:router (config-static)# maximum path ipv4 1-140000`
		Enters the maximum number of static ipv4 paths that can be configured.

Step 4	**`maximum path ipv6`** *`1-140000`*
	Example:
	`RP/0/0/CPU0:router(config-static)# maximum path ipv6 1-140000`
		Enters the maximum number of static ipv6 paths that can be configured.

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-static)# end`
	or
	`RP/0/0/CPU0:router(config-static)# 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 an IPv4 Loopback Interface

The following task describes how to configure an IPv4 Loopback interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **ipv4 address** *ipv4-address*
4. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface Loopback0`
		Enters interface configuration mode and enables a Loopback interface.

	Command or Action	Purpose
Step 3	**ipv4 address** *ipv4-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 1.1.1.1 255.255.255.255`	Enters an IPv4 address and 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 the network address.

Step 4	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 a CFI VRF Interface

Perform this task to associate a VPN routing and forwarding (VRF) instance with an interface or a subinterface on the PE routers.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address*
5. **ipv6 address** *ipv6-address*
6. **dot1q vlan** *vlan-id*
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1`	Enters interface configuration mode and enables a GigabitEthernet interface.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-if)# vrf v1`	Specifies a VRF name.
Step 4	**ipv4 address** *ipv4-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv4 address 100.1.10.2 255.255.255.0`	Enters an IPv4 address and 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.

	Command or Action	Purpose
Step 5	**ipv6 address** *ipv6-address* **Example:** `RP/0/0/CPU0:router(config-if)# ipv6 100::1:10:2/64`	Enters an IPv6 address. This argument must be in the form documented in RFC 2373, where the address is specified in hexadecimal using 16-bit values between colons, as follows: • IPv6 name or address: Hostname or X:X::X%zone • IPv6 prefix: X:X::X%zone/<0-128>
Step 6	**dot1q native vlan** *vlan-id* **Example:** `RP/0/0/CPU0:router(config-if)# dot1q native vlan 665`	(Optional) Enters the trunk interface ID. Range is from 1 to 4094 inclusive (0 and 4095 are reserved).

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 the Core Network

To configure the core network, refer to the procedures documented in *Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software.*

The tasks are presented as follows:

- Assessing the needs of MPLS VPN customers
- Configuring routing protocols in the core
- Configuring MPLS in the core
- Enabling FIB in the core
- Configuring BGP on the PE routers and route reflectors

Configuring Inter-AS and CSC Support over IP Tunnels

These tasks describe how to configure Inter-AS and CSC support over IP tunnels:

- Configuring the ASBRs to Exchange VPN-IPv4 Addresses for IP Tunnels, page VPC-247 (required)
- Configuring the Backbone Carrier Core for IP Tunnels, page VPC-250
- Configuring CSC-PE Routers for IP Tunnels, page VPC-250

Configuring the ASBRs to Exchange VPN-IPv4 Addresses for IP Tunnels

Perform this task to configure an external Border Gateway Protocol (eBGP) autonomous system boundary router (ASBR) to exchange VPN-IPv4 routes with another autonomous system for IP tunnels

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** {**ipv4 tunnel**}
4. **address-family** {**vpnv4 unicast**}
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **address-family** {**vpnv4 unicast**}
8. **route-policy** *route-policy-name* {**in**}
9. **route-policy** *route-policy-name* {**out**}
10. **neighbor** *ip-address*
11. **remote-as** *autonomous-system-number*
12. **update-source** *type interface-path-id*
13. **address-family** {**ipv4 tunnel**}
14. **address-family** {**vpnv4 unicast**}
15. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**`address-family`** `{`**`ipv4 tunnel`**`}`
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family ipv4 tunnel`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures IPv4 tunnel address family.

Step 4	**`address-family`** `{`**`vpnv4 unicast`**`}`
	Example:
	`RP/0/0/CPU0:router(cconfig-bgp-af)# address-family vpnv4 unicast`
		Configures VPNv4 address family.

Step 5	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number.

Step 7	**`address-family`** `{`**`vpnv4 unicast`**`}`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 8	**`route-policy`** *`route-policy-name`* `{`**`in`**`}`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`
		Applies a routing policy to updates that are received from a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **in** keyword to define the policy for inbound routes.

	Command or Action	Purpose

Step 9	**route-policy** *route-policy-name* {**out**}
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
	`route-policy pass-all out`
		Applies a routing policy to updates that are sent from a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **out** keyword to define the policy for outbound routes.

Step 10	**neighbor** *ip-address*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# neighbor`
	`175.40.25.2`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 175.40.25.2 as an VPNv4 iBGP peer.

Step 11	**remote-as** *autonomous-system-number*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as`
	`2002`
		Creates a neighbor and assigns it a remote autonomous system number.

Step 12	**update-source** *type interface-path-id*
	Example:
	`RP/0/0/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 13	**address-family** {**ipv4 tunnel**}
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
	`address-family ipv4 tunnel`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures IPv4 tunnel address family.

	Command or Action	Purpose

Step 14	**address-family** {**vpnv4 unicast**}
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# address-family vpnv4 unicast`
		Configures VPNv4 address family.

Step 15	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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 the Backbone Carrier Core for IP Tunnels

Configuring the backbone carrier core requires setting up connectivity and routing functions for the CSC core and the CSC-PE routers. To do so, you must complete the following high-level tasks:

- Verify IP connectivity in the CSC core.
- Configure IP tunnels in the core.
- Configure VRFs for CSC-PE routers.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

Configuring CSC-PE Routers for IP Tunnels

Perform this task to configure a CSC-PE for IP tunnels.

For information on how to configure CSC-CE routers, see the Implementing MPLS Layer 3 VPNs module.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family** {**vpnv4 unicast**}
4. **address-family** {**ipv4 tunnel**}
5. **neighbor** *A.B.C.D*
6. **remote-as** *as-number*
7. **update-source** *interface-type interface-number*
8. **address-family** {**vpnv4 unicast**}
9. **address-family** {**ipv4 tunnel**}
10. **vrf** *vrf-name*
11. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}
12. **address-family** {**ipv4 unicast**}
13. **allocate-label all**
14. **neighbor** *A.B.C.D*
15. **remote-as** *as-number*
16. **address-family** {**ipv4 labeled-unicast**}
17. **route-policy** *route-policy-name* **in**
18. **route-policy** *route-policy-name* **out**
19. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *as-number*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 2`
	`RP/0/0/CPU0:router(config-bgp)#`
		Configures a BGP routing process and enters router configuration mode.
		• Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3	**address-family** {**vpnv4 unicast**}
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

	Command or Action	Purpose
Step 4	**address-family** {**ipv4 tunnel**} **Example:** RP/0/0/CPU0:router(config-bgp-af)# address-family ipv4 tunnel	Configures IPv4 tunnel address family.
Step 5	**neighbor** *A.B.C.D* **Example:** RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.10.10.0 RP/0/0/CPU0:router(config-bgp-nbr)#	Configures the IP address for the BGP neighbor.
Step 6	**remote-as** *as-number* **Example:** RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 888	Configures the AS number for the BGP neighbor.
Step 7	**update-source** *interface-type interface-number* **Example:** RP/0/0/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 8	**address-family** {**vpnv4 unicast**} **Example:** RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast RP/0/0/CPU0:router(config-bgp-nbr-af)#	Configures VPNv4 unicast address family.
Step 9	**address-family** {**ipv4 tunnel**} **Example:** RP/0/0/CPU0:router(config-bgp-nbr-af)# address-family ipv4 tunnel	Configures IPv4 tunnel address family.
Step 10	**vrf** *vrf-name* **Example:** RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf 9999 RP/0/0/CPU0:router(config-bgp-vrf)#	Configures a VRF instance.
Step 11	**rd** {*as-number:nn* | *ip-address:nn* | **auto**} **Example:** RP/0/0/CPU0:router(config-bgp-vrf)# rd auto	Configures a route distinguisher. **Note** Use the **auto** keyword to automatically assign a unique route distinguisher.
Step 12	**address-family** {**ipv4 unicast**} **Example:** RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast RP/0/0/CPU0:router(config-bgp-vrf-af)#	Configures IPv4 unicast address family.

	Command or Action	Purpose
Step 13	**allocate-label all** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-af)# allocate-label all`	Allocate labels for all local prefixes and prefixes received with labels.

Step 14	**neighbor** *A.B.C.D*
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-af)# neighbor 10.10.10.0`
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)#`
		Configures the IP address for the BGP neighbor.

Step 15	**remote-as** *as-number* **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr)# remote-as 888`	Enables the exchange of information with a neighboring BGP router.

Step 16	**address-family** {**ipv4 labeled-unicast**}
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)#`
		Configures IPv4 labeled-unicast address family.

Step 17	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all in`	Applies the pass-all policy to all inbound routes.

	Command or Action	Purpose
Step 18	**route-policy** *route-policy-name* **out** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out`	Applies the pass-all policy to all outbound routes.

Step 19	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# 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.

Verifying MPLS VPN over IP

To verify the configuration of end-end (PE-PE) MPLS VPN over IP provisioning, use the following **show** commands:

- **show cef recursive-nexthop**
- **show bgp ipv4 tunnel**
- **show bgp vpnv4 unicast summary**
- **show bgp vrf v1 ipv4 unicast summary**
- **show bgp vrf v1 ipv4 unicast** *prefix*
- **show cef vrf v1 ipv4** *prefix*
- **show cef ipv6 recursive-nexthop**
- **show bgp vpnv6 unicast summary**
- **show bgp vrf v1 ipv6 unicast summary**
- **show bgp vrf v1 ipv6 unicast** *prefix*
- **show cef vrf v1 ipv6** *prefix*
-

Configuring Source Pool Address for MPLS VPNs over IP Tunnels

Perform this task to configure the Multiple Tunnel Source Address.

SUMMARY STEPS

1. **configure**
2. **tunnel-template** *name*
3. **mtu** *MTU value*
4. **ttl** [*ttl- value*]
5. **tos** [*tos- value*]
6. **source loopback** *type interface-path-id*
7. **source-pool** *A.B.C.D/prefix*
8. **encapsulation l2tp**
9. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**tunnel-template** *name*
	Example:
	`RP/0/0/CPU0:router(config)#tunnel-template test`
	`RP/0/0/CPU0:router(config-tuntem)#`
		Configures the tunnel template for source address.

Step 3	**mtu** [*mtu-value*]
	Example:
	`RP/0/0/CPU0:router(config-tuntem) mtu 600`
	`RP/0/0/CPU0:router(config-tuntem)#`
		Configures the maximum transmission unit for the tunnel.

Step 4	**ttl** [*ttl-value*]
	Example:
	`RP/0/0/CPU0:router(config-tuntem)ttl 64`
	`RP/0/0/CPU0:router(config-tuntem)`
		Configures the IP time to live (TTL).

Step 5	**tos** [*tos-value*]
	Example:
	`RP/0/0/CPU0:router(config-tuntem)tos 7`
	`RP/0/0/CPU0:router(config-tuntem)`
		Configures the tunnel header. By default, the TOS bits for the tunnel header are set to zero.

	Command or Action	Purpose

Step 6	**`source`** *`loopback type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config-tuntem)source loopback0`
		Configures the loopback interface.

Step 7	**`source-pool`** *`A.B.C.D/prefix`*
	Example:
	`RP/0/0/CPU0:router(config-tuntem)# source-pool 10.10.10.0/28`
		Configures the source pool address.

Step 8	**`encapsulation`** *`l2tp`*
	Example:
	`RP/0/0/CPU0:router(config-tuntem)# encapsulation l2tp`
	`RP/0/0/CPU0:router(config-config-tunencap-l2tp)#`
		Configures the Layer 2 Tunnel Protocol encapsulation.

Step 9	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-tuntem)# end`
	or
	`RP/0/0/CPU0:router(config-tuntem)# 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.

Configuration Examples for MPLS VPNs over IP Tunnels

This section provides the following examples:

- Configuring an L2TPv3 Tunnel: Example, page VPC-257
- Configuring the Global VRF Definition: Example, page VPC-257
- Configuring a Route-Policy Definition: Example, page VPC-257
- Configuring a Static Route: Example, page VPC-258
- Configuring an IPv4 Loopback Interface: Example, page VPC-258
- Configuring a CFI VRF Interface: Example, page VPC-258
- Configuring Source Pool Address for MPLS VPNs over IP Tunnels: Example, page VPC-258

Configuring an L2TPv3 Tunnel: Example

The following example shows how to configure an L2TPv3 tunnel:

```
tunnel-template t1
 encapsulation l2tp
 !
 source Loopback0
!
```

Configuring the Global VRF Definition: Example

The following example shows how to configure an L2TPv3 tunnel:

```
vrf v1
 address-family ipv4 unicast
 import route-target
   1:1
  !
  export route-target
   1:1
  !
 address-family ipv6 unicast
  import route-target
   1:1
!
  export route-target
   1:1
  !
```

Configuring a Route-Policy Definition: Example

The following example shows how to configure a route-policy definition:

```
configure
  route-policy pass-all
  pass
end-policy
!
```

Configuring a Static Route: Example

The following example shows how to configure a static route:

```
configure
  router static
  maximum path ipv4 <1-140000>
  maximum path ipv6 <1-140000>
end-policy
!
```

Configuring an IPv4 Loopback Interface: Example

The following example shows how to configure an IPv4 Loopback Interface:

```
configure
 interface Loopback0
 ipv4 address 1.1.1.1 255.255.255.255
!
```

Configuring a CFI VRF Interface: Example

The following example shows how to configure an L2TPv3 tunnel:

```
configure
  interface GigabitEthernet0/0/0/1.1
  vrf v1
   ipv4 address 100.1.10.2 255.255.255.0
   ipv6 address 100::1:10:2/64
   dot1q vlan 101
!
```

Configuring Source Pool Address for MPLS VPNs over IP Tunnels: Example

```
configure
tunnel-template test
 mtu 1500
 ttl 64
ttl 7
source Loopback0
 source-pool 10.10.10.0/28
 encapsulation l2tp
 !
```

Additional References

For additional information related to this feature, refer to the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software*
Layer 2 Tunnel Protocol Version 3	*Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software*
Routing (BGP, EIGRP, OSPF, and RIP) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*
Routing (BGP, EIGRP, OSPF, and RIP) configuration	*Cisco IOS XR Routing Configuration Guide*
MPLS LDP configuration: configuration concepts, task, and examples	*Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software*
MPLS Traffic Engineering Resource Reservation Protocol configuration: configuration concepts, task, and examples	*Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software*
Cisco CRS router getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 3931	*Layer Two Tunneling Protocol - Version 3 (L2TPv3)*
RFC 2547	*BGP/MPLS VPNs*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

Implementing MPLS Layer 3 VPNs

A Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of an MPLS provider core network. At each customer site, one or more customer edge (CE) routers attach to one or more provider edge (PE) routers.

This module provides the conceptual and configuration information for MPLS Layer 3 VPNs on Cisco IOS XR software.

Note You must acquire an evaluation or permanent license in order to use MPLS Layer 3 VPN functionality. However, if you are upgrading from a previous version of the software, MPLS Layer 3 VPN functionality will continue to work using an implicit license for 90 days (during which time, you can purchase a permanent license). For more information about licenses, see the *Software Entitlement on Cisco IOS XR Software* module in the *Cisco IOS XR System Management Configuration Guide.*

Note For a complete description of the commands listed in this module, refer to the *Cisco IOS XR MPLS Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing MPLS Layer 3 VPNs on Cisco IOS XR software

Release	Modification
Release 3.3.0	This feature was introduced.
Release 3.4.0	Support was added for MPLS L3VPN Carrier Supporting Carrier (CSC) functionality, including conceptual information and configuration tasks.
Release 3.4.1	No modification.
Release 3.5.0	Support was added for 6VPE. MPLS L3VPN Carrier Supporting Carrier (CSC) information was upgraded.
Release 3.6.0	Support was added for Inter-AS and CSC over IP Tunnels.
Release 3.7.0	Support was added for: • IPv6 VPN Provider Edge (6VPE). • Inter-AS support for 6VPE.

Contents

Prerequisites for Implementing MPLS L3VPN

The following prerequisites are required to configure MPLS Layer 3 VPN:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.

  If you need assistance with your task group assignment, contact your system administrator.
- You must be in a user group associated with a task group that includes the proper task IDs for
  - BGP commands
  - MPLS commands (generally)
  - MPLS Layer 3 VPN commands

The following prerequisites are required for configuring MPLS VPN Inter-AS with autonomous system boundary routers (ASBRs) exchanging VPN-IPV4 addresses or IPv4 routes and MPLS labels:

- Before configuring external Border Gateway Protocol (eBGP) routing between autonomous systems or subautonomous systems in an MPLS VPN, ensure that all MPLS VPN routing instances and sessions are properly configured (see the How to Implement MPLS Layer 3 VPNs, page VPC-285 for procedures).
- The following tasks must be performed:
  - Define VPN routing instances
  - Configure BGP routing sessions in the MPLS core
  - Configure PE-to-PE routing sessions in the MPLS core
  - Configure BGP PE-to-CE routing sessions
  - Configure a VPN-IPv4 eBGP session between directly connected ASBRs

To configure MPLS Layer 3 VPNs, routers must support MPLS forwarding and Forwarding Information Base (FIB).

MPLS L3VPN Restrictions

The following are restrictions for implementing MPLS Layer 3 VPNs:

- Multihop VPN-IPv4 eBGP is not supported for configuring eBGP routing between autonomous systems or subautonomous systems in an MPLS VPN.
- MPLS VPN supports only IPv4 address families.

The following restrictions apply when configuring MPLS VPN Inter-AS with ASBRs exchanging IPv4 routes and MPLS labels:

- For networks configured with eBGP multihop, a label switched path (LSP) must be configured between nonadjacent routers.
- Inter-AS supports IPv4 routes only. IPv6 is not supported.

Note The physical interfaces that connect the BGP speakers must support FIB and MPLS.

The following restrictions apply to routing protocols OSPF and RIP:

- IPv6 is not supported on OSPF and RIP.

Information About MPLS Layer 3 VPNs

To implement MPLS Layer 3 VPNs, you need to understand the following concepts:

MPLS L3VPN Overview

Before defining an MPLS VPN, VPN in general must be defined. A VPN is:

- An IP-based network delivering private network services over a public infrastructure
- A set of sites that are allowed to communicate with each other privately over the Internet or other public or private networks

Conventional VPNs are created by configuring a full mesh of tunnels or permanent virtual circuits (PVCs) to all sites in a VPN. This type of VPN is not easy to maintain or expand, as adding a new site requires changing each edge device in the VPN.

MPLS-based VPNs are created in Layer 3 and are based on the peer model. The peer model enables the service provider and the customer to exchange Layer 3 routing information. The service provider relays the data between the customer sites without customer involvement.

MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the edge router of the service provider that provides services to the customer site needs to be updated.

The components of the MPLS VPN are described as follows:

- Provider (P) router—Router in the core of the provider network. PE routers run MPLS switching and do not attach VPN labels to routed packets. VPN labels are used to direct data packets to the correct private network or customer edge router.
- PE router—Router that attaches the VPN label to incoming packets based on the interface or subinterface on which they are received, and also attaches the MPLS core labels. A PE router attaches directly to a CE router.
- Customer (C) router—Router in the Internet service provider (ISP) or enterprise network.
- Customer edge (CE) router—Edge router on the network of the ISP that connects to the PE router on the network. A CE router must interface with a PE router.

Figure 25 shows a basic MPLS VPN topology.

Figure 25 Basic MPLS VPN Topology

MPLS L3VPN Benefits

MPLS L3VPN provides the following benefits:

- Service providers can deploy scalable VPNs and deliver value-added services.
- Connectionless service guarantees that no prior action is necessary to establish communication between hosts.
- Centralized Service: Building VPNs in Layer 3 permits delivery of targeted services to a group of users represented by a VPN.
- Scalability: Create scalable VPNs using connection-oriented, point-to-point overlays, Frame Relay, or ATM virtual connections.
- Security: Security is provided at the edge of a provider network (ensuring that packets received from a customer are placed on the correct VPN) and in the backbone.
- Integrated Quality of Service (QoS) support: QoS provides the ability to address predictable performance and policy implementation and support for multiple levels of service in an MPLS VPN.
- Straightforward Migration: Service providers can deploy VPN services using a straightforward migration path.

- Migration for the end customer is simplified. There is no requirement to support MPLS on the CE router and no modifications are required for a customer intranet.

How MPLS L3VPN Works

MPLS VPN functionality is enabled at the edge of an MPLS network. The PE router performs the following tasks:

- Exchanges routing updates with the CE router
- Translates the CE routing information into VPN version 4 (VPNv4) and VPN version 6 (VPNv6) routes
- Exchanges VPNv4 and VPNv6 routes with other PE routers through the Multiprotocol Border Gateway Protocol (MP-BGP)

Virtual Routing and Forwarding Tables

Each VPN is associated with one or more VPN routing and forwarding (VRF) instances. A VRF defines the VPN membership of a customer site attached to a PE router. A VRF consists of the following components:

- An IP version 4 (IPv4) unicast routing table
- A derived FIB table
- A set of interfaces that use the forwarding table
- A set of rules and routing protocol parameters that control the information that is included in the routing table

These components are collectively called a VRF instance.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A site can be a member of multiple VPNs. However, a site can associate with only one VRF. A VRF contains all the routes available to the site from the VPNs of which it is a member.

Packet forwarding information is stored in the IP routing table and the FIB table for each VRF. A separate set of routing and FIB tables is maintained for each VRF. These tables prevent information from being forwarded outside a VPN and also prevent packets that are outside a VPN from being forwarded to a router within the VPN.

VPN Routing Information: Distribution

The distribution of VPN routing information is controlled through the use of VPN route target communities, implemented by BGP extended communities. VPN routing information is distributed as follows:

- When a VPN route that is learned from a CE router is injected into a BGP, a list of VPN route target extended community attributes is associated with it. Typically, the list of route target community extended values is set from an export list of route targets associated with the VRF from which the route was learned.
- An import list of route target extended communities is associated with each VRF. The import list defines route target extended community attributes that a route must have for the route to be imported into the VRF. For example, if the import list for a particular VRF includes route target extended communities A, B, and C, then any VPN route that carries any of those route target extended communities—A, B, or C—is imported into the VRF.

BGP Distribution of VPN Routing Information

A PE router can learn an IP prefix from the following sources:

- A CE router by static configuration
- An eBGP session with the CE router
- A Routing Information Protocol (RIP) exchange with the CE router
- Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and RIP as Interior Gateway Protocols (IGPs)

The IP prefix is a member of the IPv4 address family. After the PE router learns the IP prefix, the PE converts it into the VPN-IPv4 prefix by combining it with a 64-bit route distinguisher. The generated prefix is a member of the VPN-IPv4 address family. It uniquely identifies the customer address, even if the customer site is using globally nonunique (unregistered private) IP addresses. The route distinguisher used to generate the VPN-IPv4 prefix is specified by the **rd** command associated with the VRF on the PE router.

BGP distributes reachability information for VPN-IPv4 prefixes for each VPN. BGP communication takes place at two levels:

- Within the IP domain, known as an autonomous system.
- Between autonomous systems.

PE to PE or PE to route reflector (RR) sessions are iBGP sessions, and PE to CE sessions are eBGP sessions. PE to CE eBGP sessions can be directly or indirectly connected (eBGP multihop).

BGP propagates reachability information for VPN-IPv4 prefixes among PE routers by the BGP protocol extensions (see RFC 2283, Multiprotocol Extensions for BGP-4), which define support for address families other than IPv4. Using the extensions ensures that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.

MPLS Forwarding

Based on routing information stored in the VRF IP routing table and the VRF FIB table, packets are forwarded to their destination using MPLS.

A PE router binds a label to each customer prefix learned from a CE router and includes the label in the network reachability information for the prefix that it advertises to other PE routers. When a PE router forwards a packet received from a CE router across the provider network, it labels the packet with the label learned from the destination PE router. When the destination PE router receives the labeled packet, it pops the label and uses it to direct the packet to the correct CE router. Label forwarding across the provider backbone is based on either dynamic label switching or traffic engineered paths. A customer data packet carries two levels of labels when traversing the backbone:

- The top label directs the packet to the correct PE router.
- The second label indicates how that PE router should forward the packet to the CE router.

More labels can be stacked if other features are enabled. For example, if traffic engineering (TE) tunnels with fast reroute (FRR) are enabled, the total number of labels imposed in the PE is four (Layer 3 VPN, Label Distribution Protocol (LDP), TE, and FRR).

Automatic Route Distinguisher Assignment

To take advantage of iBGP load balancing, every network VRF must be assigned a unique route distinguisher. VRFs require a route distinguisher for BGP to distinguish between potentially identical prefixes received from different VPNs.

With thousands of routers in a network each supporting multiple VRFs, configuration and management of route distinguishers across the network can present a problem. Cisco IOS XR software simplifies this process by assigning unique route distinguisher to VRFs using the **rd auto** command.

To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across the VRFs.

Finally, route distinguisher values are checkpointed so that route distinguisher assignment to VRF is persistent across failover or process restart. If an route distinguisher is explicitely configured for a VRF, this value is not overridden by the autoroute distinguisher.

MPLS L3VPN Major Components

An MPLS-based VPN network has three major components:

- VPN route target communities—A VPN route target community is a list of all members of a VPN community. VPN route targets need to be configured for each VPN community member.
- Multiprotocol BGP (MP-BGP) peering of the VPN community PE routers—MP-BGP propagates VRF reachability information to all members of a VPN community. MP-BGP peering needs to be configured in all PE routers within a VPN community.
- MPLS forwarding—MPLS transports all traffic between all VPN community members across a VPN service-provider network.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A given site can be a member of multiple VPNs. However, a site can associate with only one VRF. A customer-site VRF contains all the routes available to the site from the VPNs of which it is a member.

Inter-AS Support for L3VPN

This section contains the following topics:

Inter-AS Restrictions

Inter-AS functionality is available using VPNv4 only. VPNv6 is not currently supported.

Inter-AS Support: Overview

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

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

An MPLS VPN Inter-AS provides the following benefits:

- Allows a VPN to cross more than one service provider backbone.

  Service providers, running separate autonomous systems, can jointly offer MPLS VPN services to the same end customer. A VPN can begin at one customer site and traverse different VPN service provider backbones before arriving at another site of the same customer. Previously, MPLS VPN could traverse only a single BGP autonomous system service provider backbone. This feature lets multiple autonomous systems form a continuous, seamless network between customer sites of a service provider.

- Allows a VPN to exist in different areas.

  A service provider can create a VPN in different geographic areas. Having all VPN traffic flow through one point (between the areas) allows for better rate control of network traffic between the areas.

- Allows confederations to optimize iBGP meshing.

  Internal Border Gateway Protocol (iBGP) meshing in an autonomous system is more organized and manageable. You can divide an autonomous system into multiple, separate subautonomous systems and then classify them into a single confederation. This capability lets a service provider offer MPLS VPNs across the confederation, as it supports the exchange of labeled VPN-IPv4 Network Layer Reachability Information (NLRI) between the subautonomous systems that form the confederation.

Inter-AS and ASBRs

Separate autonomous systems from different service providers can communicate by exchanging IPv4 NLRI in the form of VPN-IPv4 addresses. The ASBRs use eBGP to exchange that information. Then an Interior Gateway Protocol (IGP) distributes the network layer information for VPN-IPV4 prefixes throughout each VPN and each autonomous system. The following protocols are used for sharing routing information:

- Within an autonomous system, routing information is shared using an IGP.
- Between autonomous systems, routing information is shared using an eBGP. An eBGP lets service providers set up an interdomain routing system that guarantees the loop-free exchange of routing information between separate autonomous systems.

The primary function of an eBGP is to exchange network reachability information between autonomous systems, including information about the list of autonomous system routes. The autonomous systems use EBGP border edge routers to distribute the routes, which include label switching information. Each border edge router rewrites the next-hop and MPLS labels.

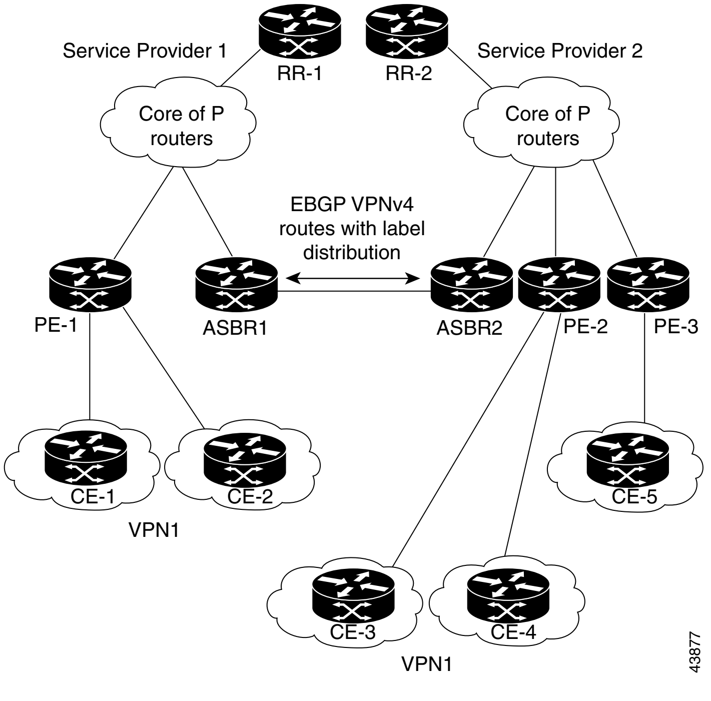
Inter-AS configurations supported in an MPLS VPN can include:

- Interprovider VPN—MPLS VPNs that include two or more autonomous systems, connected by separate border edge routers. The autonomous systems exchange routes using eBGP. No IGP or routing information is exchanged between the autonomous systems.
- BGP Confederations—MPLS VPNs that divide a single autonomous system into multiple subautonomous systems and classify them as a single, designated confederation. The network recognizes the confederation as a single autonomous system. The peers in the different autonomous systems communicate over eBGP sessions; however, they can exchange route information as if they were iBGP peers.

Transmitting Information Between Autonomous Systems

Figure 26 illustrates one MPLS VPN consisting of two separate autonomous systems. Each autonomous system operates under different administrative control and runs a different IGP. Service providers exchange routing information through eBGP border edge routers (ABSR1 and ASBR2).

Figure 26 eBGP Connection Between Two MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Addresses



This configuration uses the following process to transmit information:

Step 1 The provider edge router (PE-1) assigns a label for a route before distributing that route. The PE router uses the multiprotocol extensions of BGP to transmit label mapping information. The PE router distributes the route as a VPN-IPv4 address. The address label and the VPN identifier are encoded as part of the NLRI.

Step 2 The two route reflectors (RR-1 and RR-2) reflect VPN-IPv4 internal routes within the autonomous system. The border edge routers of the autonomous system (ASBR1 and ASBR2) advertise the VPN-IPv4 external routes.

Step 3 The eBGP border edge router (ASBR1) redistributes the route to the next autonomous system (ASBR2). ASBR1 specifies its own address as the value of the eBGP next-hop attribute and assigns a new label. The address ensures:

- That the next-hop router is always reachable in the service provider (P) backbone network.
- That the label assigned by the distributing router is properly interpreted. (The label associated with a route must be assigned by the corresponding next-hop router.)

Step 4 The eBGP border edge router (ASBR2) redistributes the route in one of the following ways, depending on the configuration:

- If the iBGP neighbors are configured with the **next-hop-self** command, ASBR2 changes the next-hop address of updates received from the eBGP peer, then forwards it.
- If the iBGP neighbors are not configured with the **next-hop-self** command, the next-hop address remains unchanged. ASBR2 must propagate a host route for the eBGP peer through the IGP. To propagate the eBGP VPN-IPv4 neighbor host route, use the **redistribute** command with the **static** keyword. An eBGP VPN-IPv4 neighbor host route must be manually configured to establish the label switched path (LSP) towards ASBR1. The static route needs to be redistributed to IGP, to let other PE routers use the /32 host prefix label to forward traffic for an Inter-AS VPN redistribute static option.

Note This option is not supported for Inter-AS over IP tunnels.

Exchanging VPN Routing Information

Autonomous systems exchange VPN routing information (routes and labels) to establish connections. To control connections between autonomous systems, the PE routers and eBGP border edge routers maintain a label forwarding information base (LFIB). The LFIB manages the labels and routes that the PE routers and eBGP border edge routers receive during the exchange of VPN information.

The autonomous systems use the following guidelines to exchange VPN routing information:

- Routing information includes:
  - The destination network (N)
  - The next-hop field associated with the distributing router
  - A local MPLS label (L)
- A route distinguisher (RD1). A route distinguisher is part of a destination network address. It makes the VPN-IPv4 route globally unique in the VPN service provider environment.

- The ASBRs are configured to change the next-hop when sending VPN-IPv4 NLRIs to the iBGP neighbors. Therefore, the ASBRs must allocate a new label when they forward the NLRI to the iBGP neighbors.

Figure 27 Exchanging Routes and Labels Between MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Address

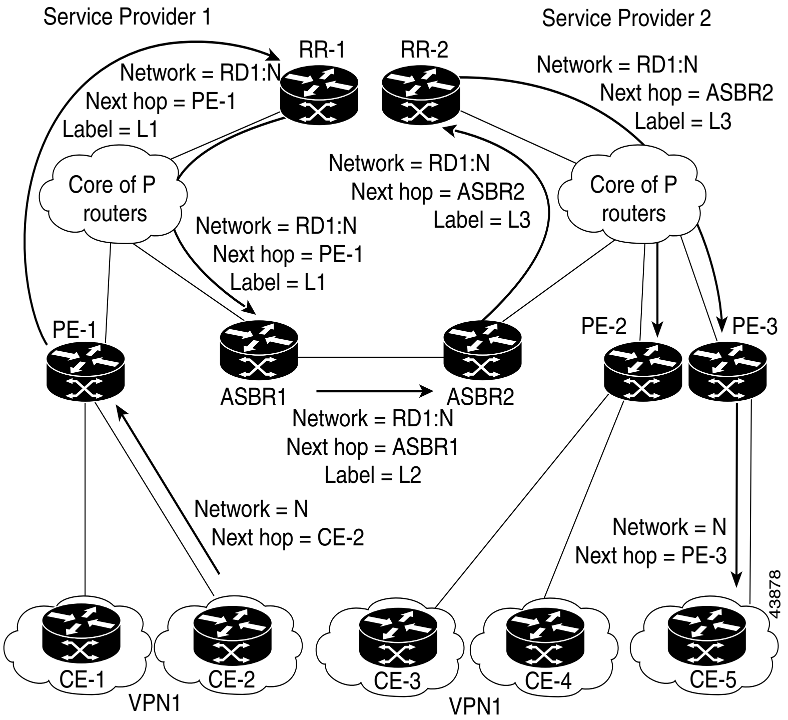
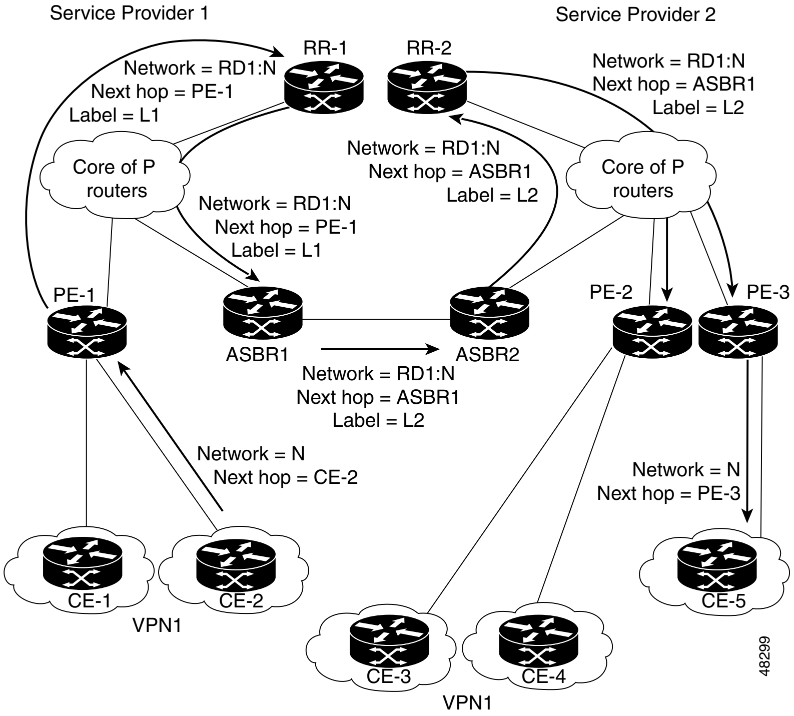


Figure 28 illustrates the exchange of VPN route and label information between autonomous systems. The only difference is that ASBR2 is configured with the **redistribute** command with the **connected** keyword, which propagates the host routes to all PEs. The command is necessary as ASBR2 is not configured to change the next-hop address.

Note Figure 28 is not applicable to Inter-AS over IP tunnels.

Figure 28 Exchanging Routes and Labels with the redistributed Command in an MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses



Packet Forwarding

Note This section is not applicable to Inter-AS over IP tunnels.

Figure 29 illustrates how packets are forwarded between autonomous systems in an interprovider network using the following packet method.

Packets are forwarded to their destination by means of MPLS. Packets use the routing information stored in the LFIB of each PE router and eBGP border edge router.

The service provider VPN backbone uses dynamic label switching to forward labels.

Each autonomous system uses standard multilevel labeling to forward packets between the edges of the autonomous system routers (for example, from CE-5 to PE-3). Between autonomous systems, only a single level of labeling is used, corresponding to the advertised route.

A data packet carries two levels of labels when traversing the VPN backbone:

- The first label (IGP route label) directs the packet to the correct PE router on the eBGP border edge router. (For example, the IGP label of ASBR2 points to the ASBR2 border edge router.)
- The second label (VPN route label) directs the packet to the appropriate PE router or eBGP border edge router.

Figure 29 Forwarding Packets Between MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Addresses

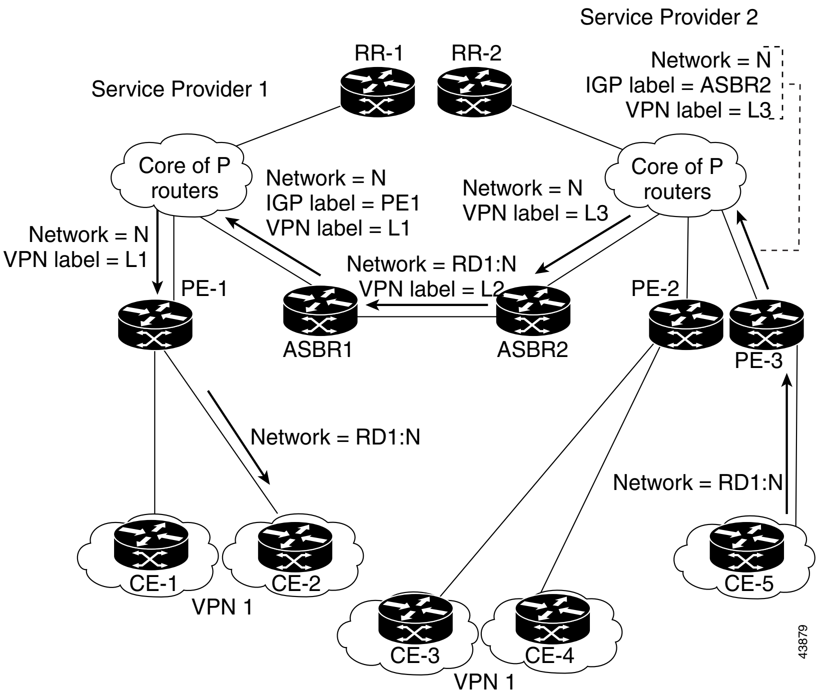


Figure 30 shows the same packet forwarding method, except the eBGP router (ASBR1) forwards the packet without reassigning a new label to it.

Figure 30 Forwarding Packets Without a New Label Assignment Between MPLS VPN Inter-AS System with ASBRs Exchanging VPN-IPv4 Addresses

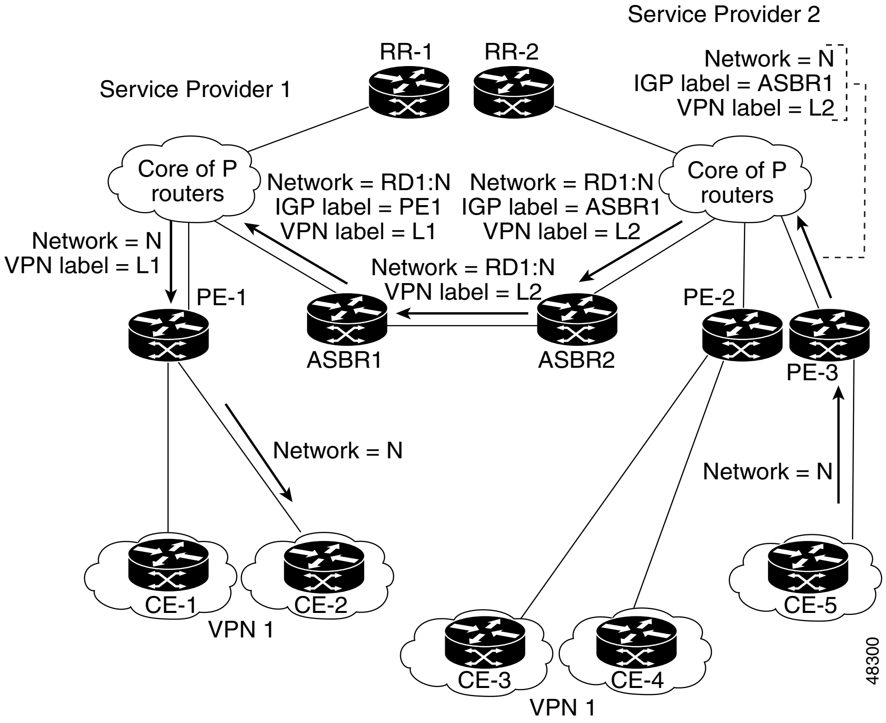
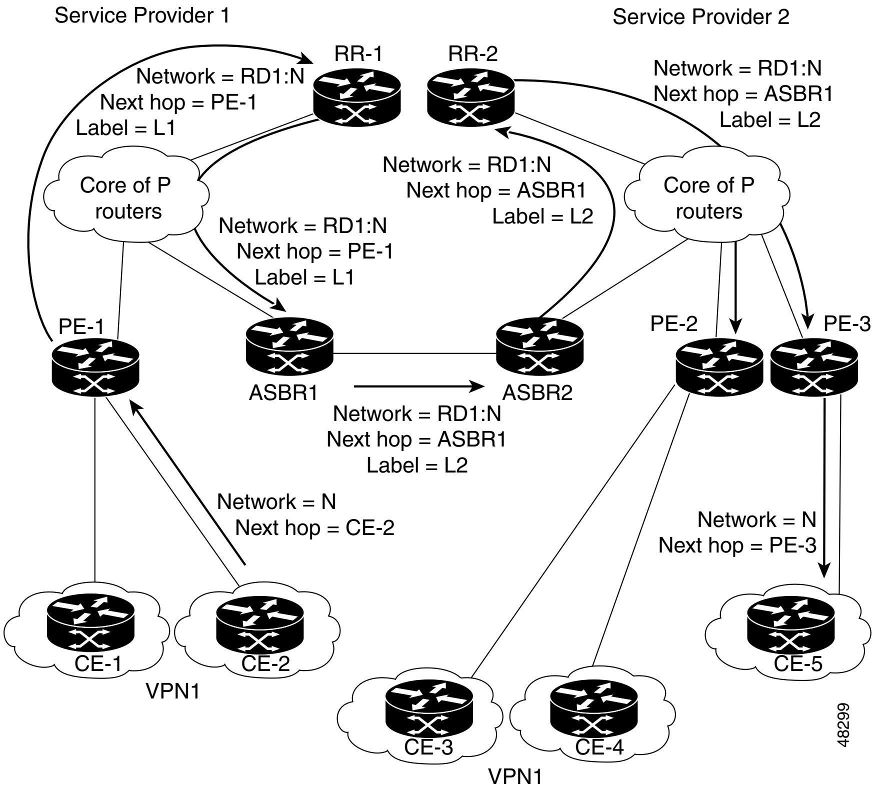


Figure 31 illustrates the exchange of VPN route and label information between autonomous systems.

Figure 31 Exchanging Routes and Labels in an MPLS VPN Inter-AS with ASBRs



Confederations

A confederation is multiple subautonomous systems grouped together. A confederation reduces the total number of peer devices in an autonomous system. A confederation divides an autonomous system into subautonomous systems and assigns a confederation identifier to the autonomous systems. A VPN can span service providers running in separate autonomous systems or multiple subautonomous systems that form a confederation.

In a confederation, each subautonomous system is fully meshed with other subautonomous systems. The subautonomous systems communicate using an IGP, such as Open Shortest Path First (OSPF) or Intermediate System-to-Intermediate System (IS-IS). Each subautonomous system also has an eBGP connection to the other subautonomous systems. The confederation eBGP (CEBGP) border edge routers forward next-hop-self addresses between the specified subautonomous systems. The next-hop-self address forces the BGP to use a specified address as the next hop rather than letting the protocol choose the next hop.

You can configure a confederation with separate subautonomous systems two ways:

- Configure a router to forward next-hop-self addresses between only the CEBGP border edge routers (both directions). The subautonomous systems (iBGP peers) at the subautonomous system border do not forward the next-hop-self address. Each subautonomous system runs as a single IGP domain. However, the CEBGP border edge router addresses are known in the IGP domains.

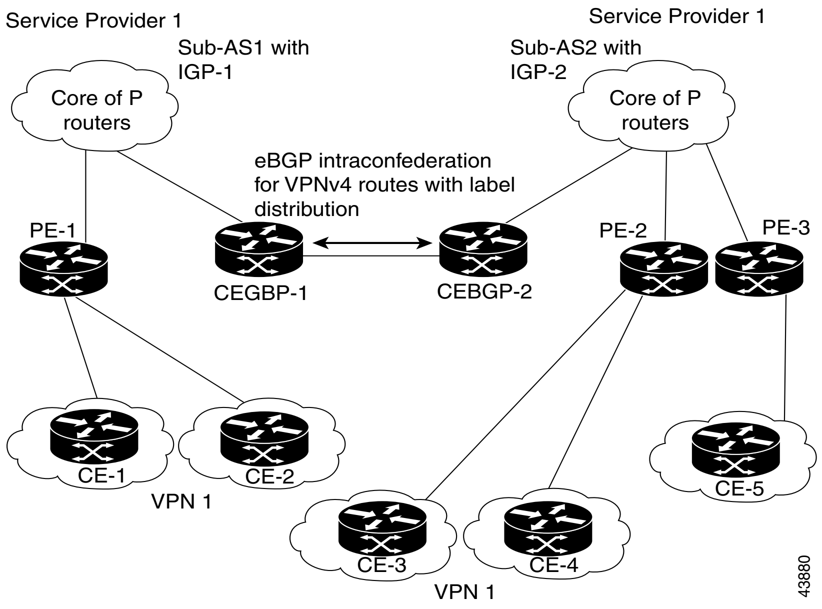
- Configure a router to forward next-hop-self addresses between the CEBGP border edge routers (both directions) and within the iBGP peers at the subautonomous system border. Each subautonomous system runs as a single IGP domain but also forwards next-hop-self addresses between the PE routers in the domain. The CEBGP border edge router addresses are known in the IGP domains.

Note Figure 26 illustrates how two autonomous systems exchange routes and forward packets. Subautonomous systems in a confederation use a similar method of exchanging routes and forwarding packets.

Figure 32 illustrates a typical MPLS VPN confederation configuration. In this configuration:

- The two CEBGP border edge routers exchange VPN-IPv4 addresses with labels between the two autonomous systems.
- The distributing router changes the next-hop addresses and labels and uses a next-hop-self address.
- IGP-1 and IGP-2 know the addresses of CEBGP-1 and CEBGP-2.

Figure 32 eBGP Connection Between Two Subautonomous Systems in a Confederation



In this confederation configuration:

- CEBGP border edge routers function as neighboring peers between the subautonomous systems. The subautonomous systems use eBGP to exchange route information.
- Each CEBGP border edge router (CEBGP-1 and CEBGP-2) assigns a label for the router before distributing the route to the next subautonomous system. The CEBGP border edge router distributes the route as a VPN-IPv4 address by using the multiprotocol extensions of BGP. The label and the VPN identifier are encoded as part of the NLRI.

- Each PE and CEBGP border edge router assigns its own label to each VPN-IPv4 address prefix before redistributing the routes. The CEBGP border edge routers exchange IPV-IPv4 addresses with the labels. The next-hop-self address is included in the label (as the value of the eBGP next-hop attribute). Within the subautonomous systems, the CEBGP border edge router address is distributed throughout the iBGP neighbors, and the two CEBGP border edge routers are known to both confederations.

For more information about how to configure confederations, see the "Configuring MPLS Forwarding for ASBR Confederations" section on page MPC-325.

MPLS VPN Inter-AS BGP Label Distribution

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up the MPLS VPN Inter-AS network so that the ASBRs exchange IPv4 routes with MPLS labels of the provider edge (PE) routers. Route reflectors (RRs) exchange VPN-IPv4 routes by using multihop, multiprotocol external Border Gateway Protocol (eBGP). This method of configuring the Inter-AS system is often called MPLS VPN Inter-AS BGP Label Distribution.

Configuring the Inter-AS system so that the ASBRs exchange the IPv4 routes and MPLS labels has the following benefits:

- Saves the ASBRs from having to store all the VPN-IPv4 routes. Using the route reflectors to store the VPN-IPv4 routes and forward them to the PE routers results in improved scalability compared with configurations in which the ASBR holds all the VPN-IPv4 routes and forwards the routes based on VPN-IPv4 labels.
- Having the route reflectors hold the VPN-IPv4 routes also simplifies the configuration at the border of the network.
- Enables a non-VPN core network to act as a transit network for VPN traffic. You can transport IPv4 routes with MPLS labels over a non-MPLS VPN service provider.
- Eliminates the need for any other label distribution protocol between adjacent label switch routers (LSRs). If two adjacent LSRs are also BGP peers, BGP can handle the distribution of the MPLS labels. No other label distribution protocol is needed between the two LSRs.

Exchanging IPv4 Routes with MPLS labels

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up a VPN service provider network to exchange IPv4 routes with MPLS labels. You can configure the VPN service provider network as follows:

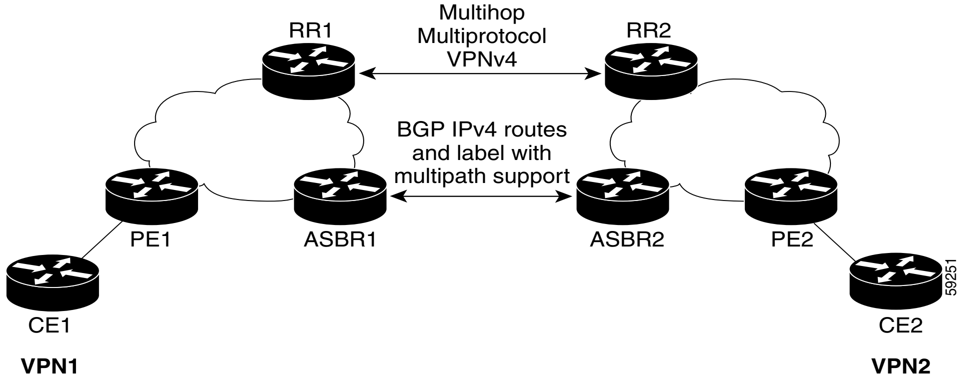
- Route reflectors exchange VPN-IPv4 routes by using multihop, multiprotocol eBGP. This configuration also preserves the next-hop information and the VPN labels across the autonomous systems.
- A local PE router (for example, PE1 in Figure 33) needs to know the routes and label information for the remote PE router (PE2).

This information can be exchanged between the PE routers and ASBRs in one of two ways:

- Internal Gateway Protocol (IGP) and Label Distribution Protocol (LDP): The ASBR can redistribute the IPv4 routes and MPLS labels it learned from eBGP into IGP and LDP and from IGP and LDP into eBGP.
- Internal Border Gateway Protocol (iBGP) IPv4 label distribution: The ASBR and PE router can use direct iBGP sessions to exchange VPN-IPv4 and IPv4 routes and MPLS labels.

  Alternatively, the route reflector can reflect the IPv4 routes and MPLS labels learned from the ASBR to the PE routers in the VPN. This reflecting of learned IPv4 routes and MPLS labels is accomplished by enabling the ASBR to exchange IPv4 routes and MPLS labels with the route reflector. The route reflector also reflects the VPN-IPv4 routes to the PE routers in the VPN. For example, in VPN1, RR1 reflects to PE1 the VPN-IPv4 routes it learned and IPv4 routes and MPLS labels learned from ASBR1. Using the route reflectors to store the VPN-IPv4 routes and forward them through the PE routers and ASBRs allows for a scalable configuration.

Figure 33 VPNs Using eBGP and iBGP to Distribute Routes and MPLS Labels



BGP Routing Information

BGP routing information includes the following items:

- Network number (prefix), which is the IP address of the destination.
- Autonomous system (AS) path, which is a list of the other ASs through which a route passes on the way to the local router. The first AS in the list is closest to the local router; the last AS in the list is farthest from the local router and usually the AS where the route began.
- Path attributes, which provide other information about the AS path, for example, the next hop.

BGP Messages and MPLS Labels

MPLS labels are included in the update messages that a router sends. Routers exchange the following types of BGP messages:

- Open messages—After a router establishes a TCP connection with a neighboring router, the routers exchange open messages. This message contains the number of the autonomous system to which the router belongs and the IP address of the router that sent the message.
- Update messages—When a router has a new, changed, or broken route, it sends an update message to the neighboring router. This message contains the NLRI, which lists the IP addresses of the usable routes. The update message includes any routes that are no longer usable. The update message also includes path attributes and the lengths of both the usable and unusable paths. Labels for VPN-IPv4 routes are encoded in the update message, as specified in RFC 2858. The labels for the IPv4 routes are encoded in the update message, as specified in RFC 3107.
- Keepalive messages—Routers exchange keepalive messages to determine if a neighboring router is still available to exchange routing information. The router sends these messages at regular intervals. (Sixty seconds is the default for Cisco routers.) The keepalive message does not contain routing data; it contains only a message header.
- Notification messages—When a router detects an error, it sends a notification message.

Sending MPLS Labels with Routes

When BGP (eBGP and iBGP) distributes a route, it can also distribute an MPLS label that is mapped to that route. The MPLS label mapping information for the route is carried in the BGP update message that contains the information about the route. If the next hop is not changed, the label is preserved.

When you issue the **show bgp neighbors** *ip-address* command on both BGP routers, the routers advertise to each other that they can then send MPLS labels with the routes. If the routers successfully negotiate their ability to send MPLS labels, the routers add MPLS labels to all outgoing BGP updates.

Carrier Supporting Carrier Support for L3VPN

This section provides conceptual information about MPLS VPN Carrier Supporting Carrier (CSC) functionality and includes the following topics:

- CSC Prerequisites, page VPC-280
- CSC Benefits, page VPC-280
- Configuration Options for the Backbone and Customer Carriers, page VPC-281

Throughout this document, the following terminology is used in the context of CSC:

backbone carrier—Service provider that provides the segment of the backbone network to the other provider. A backbone carrier offers BGP and MPLS VPN services.

customer carrier—Service provider that uses the segment of the backbone network. The customer carrier may be an Internet service provider (ISP) or a BGP/MPLS VPN service provider.

CE router—A customer edge router is part of a customer network and interfaces to a provider edge (PE) router. In this document, the CE router sits on the edge of the customer carrier network.

PE router—A provider edge router is part of a service provider's network connected to a customer edge (CE) router. In this document, the PE router sits on the edge of the backbone carrier network

ASBR—An autonomous system boundary router connects one autonomous system to another.

CSC Prerequisites

The following prerequisites are required to configure CSC:

- You must be able to configure MPLS VPNs with end-to-end (CE-to-CE router) pings working.
- You must be able to configure Interior Gateway Protocols (IGPs), MPLS Label Distribution Protocol (LDP), and Multiprotocol Border Gateway Protocol (MP-BGP).
- You must ensure that CSC-PE and CSC-CE routers support BGP label distribution.

Note BGP is the only supported label distribution protocol on the link between CE and PE.

CSC Benefits

This section describes the benefits of CSC to the backbone carrier and customer carriers.

Benefits to the Backbone Carrier

- The backbone carrier can accommodate many customer carriers and give them access to its backbone.
- The MPLS VPN carrier supporting carrier feature is scalable.
- The MPLS VPN carrier supporting carrier feature is a flexible solution.

Benefits to the Customer Carriers

- The MPLS VPN carrier supporting carrier feature removes from the customer carrier the burden of configuring, operating, and maintaining its own backbone.

- Customer carriers who use the VPN services provided by the backbone carrier receive the same level of security that Frame Relay or ATM-based VPNs provide.
- Customer carriers can use any link layer technology to connect the CE routers to the PE routers and the PE routers to the P routers.
- The customer carrier can use any addressing scheme and still be supported by a backbone carrier.

Benefits of Implementing MPLS VPN CSC Using BGP

The benefits of using BGP to distribute IPv4 routes and MPLS label routes are:

- BGP takes the place of an IGP and LDP in a VPN forwarding and routing instance (VRF) table.
- BGP is the preferred routing protocol for connecting two ISPs,

Configuration Options for the Backbone and Customer Carriers

To enable CSC, the backbone and customer carriers must be configured accordingly:

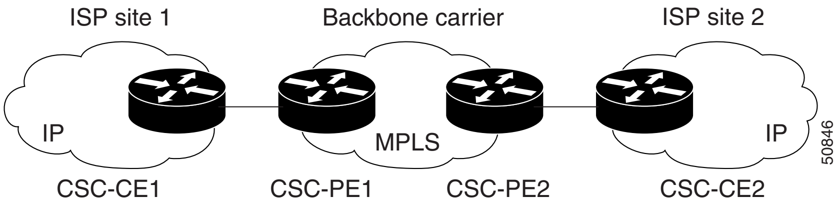
- The backbone carrier must offer BGP and MPLS VPN services.
- The customer carrier can take several networking forms. The customer carrier can be:
  - An ISP with an IP core (see the "Customer Carrier: ISP with IP Core" section on page MPC-281).
  - An MPLS service provider with or without VPN services (see "Customer Carrier: MPLS Service Provider" section on page MPC-282).

Note An IGP in the customer carrier network is used to distribute next hops and loopbacks to the CSC-CE. IBGP with label sessions are used in the customer carrier network to distribute next hops and loopbacks to the CSC-CE.

Customer Carrier: ISP with IP Core

Figure 34 shows a network configuration where the customer carrier is an ISP. The customer carrier has two sites, each of which is a point of presence (POP). The customer carrier connects these sites using a VPN service provided by the backbone carrier. The backbone carrier uses MPLS or IP tunnels to provide VPN services. The ISP sites use IP.

Figure 34 Network: Customer Carrier Is an ISP

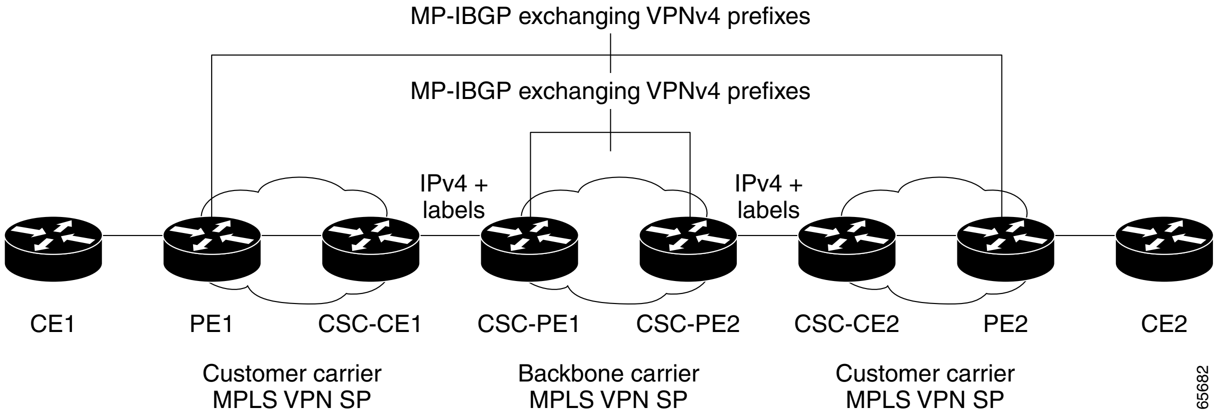


The links between the CE and PE routers use eBGP to distribute IPv4 routes and MPLS labels. Between the links, the PE routers use multiprotocol iBGP to distribute VPNv4 routes.

Customer Carrier: MPLS Service Provider

Figure 35 shows a network configuration where the backbone carrier and the customer carrier are BGP/MPLS VPN service providers. The customer carrier has two sites. The customer carrier uses MPLS in its network while the backbone carrier may use MPLS or IP tunnels in its network.

Figure 35 Network: Customer Carrier Is an MPLS VPN Service Provider


MP-IBGP exchanging VPNv4 prefixes
MP-IBGP exchanging VPNv4 prefixes
IPv4 + labels
IPv4 + labels
CE1
PE1
CSC-CE1
CSC-PE1
CSC-PE2
CSC-CE2
PE2
CE2
Customer carrier MPLS VPN SP
Backbone carrier MPLS VPN SP
Customer carrier MPLS VPN SP
65682

In this configuration (Figure 35), the customer carrier can configure its network in one of these ways:

- The customer carrier can run an IGP and LDP in its core network. In this case, the CSC-CE1 router in the customer carrier redistributes the eBGP routes it learns from the CSC-PE1 router of the backbone carrier to an IGP.
- The CSC-CE1 router of the customer carrier system can run an IPv4 and labels iBGP session with the PE1 router.

IPv6 VPN Provider Edge (6VPE) Support

6VPE uses the existing MPLS IPv4 core infrastructure for IPv6 transports to enable IPv6 sites to communicate over an MPLS IPv4 core network using MPLS label switch paths (LSPs). 6VPE relies on multiprotocol BGP extensions in the IPv4 network configuration on the provider edge (PE) router to exchange IPv6 reachability information. Edge routers are then configured to be dual stacks running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange (see “Dual Stack” section on page MPC-283).

This section includes the follow subsections:

- 6VPE Benefits, page VPC-283
- 6VPE Network Architecture, page VPC-283
- Dual Stack, page VPC-283
- 6VPE Operation, page VPC-284

6VPE Benefits

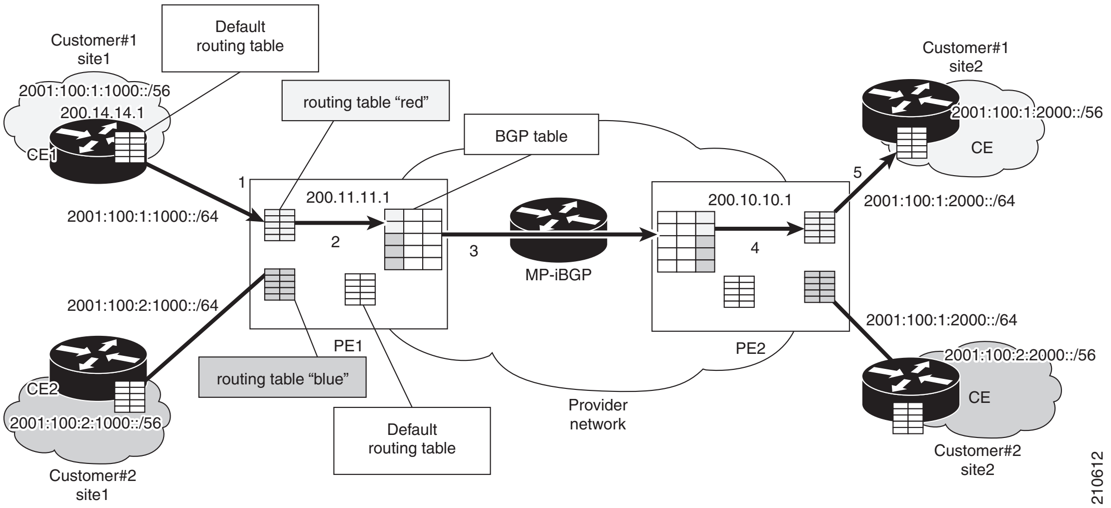
6VPE provides the following benefits to service providers:

- Support for IPv6 without changing the IPv4 MPLS backbone.
- No requirement for a separate signaling plane.
- Leverages operational IPv4 MPLS backbones.
- Cost savings from operating expenses.
- Addresses the security limitations of 6PE.
- Provides logically-separate routing table entries for VPN member devices.
- Provides support for Inter-AS and CSC scenarios. Inter-AS support for 6VPE requires support of Border Gateway Protocol (BGP) to enable the address families and to allocate and distribute the PE and ASBR labels.

6VPE Network Architecture

Figure 36 illustrates the 6VPE network architecture and control plane protocols when two IPv6 sites communicate through an MPLSv4 backbone.

Figure 36 6VPE Network Architecture



Dual Stack

Dual stack is a technique that lets IPv4 and IPv6 coexist on the same interfaces. Coexistence of IPv4 and IPv6 is a requirement for initial deployment. With regard to supporting IPv6 on a MPLS network, two important aspects of the network should be reviewed:

- *Core*: The 6VPE technique carries IPv6 in a VPN fashion over a non-IPv6-aware MPLS core, and enables IPv4 or IPv6 communities to communicate with each other over an IPv4 MPLS backbone without modifying the core infrastructure. By avoiding dual stacking on the core routers, the resources can be dedicated to their primary function to avoid any complexity on the operational side. The transition and integration with respect to the current state of networks is also transparent.
- *Access*: To support native IPv6, the access that connects to IPv4 and IPv6 domains must be IPv6-aware. Service provider edge elements can exchange routing information with end users; therefore, dual stacking is a mandatory requirement on the access layer.

6VPE Operation

When IPv6 is enabled on the subinterface that is participating in a VPN, it becomes an IPv6 VPN. The customer edge-provider edge link is running IPv6 or IPv4 natively. The addition of IPv6 on a provider edge router turns the provider edge into 6VPE, thereby enabling service providers to support IPv6 over the MPLS network.

Provider edge routers use VRF tables to maintain the segregated reachability and forwarding information of each IPv6 VPN. MPBGP with its IPv6 extensions distributes the routes from 6VPE to other 6VPEs through a direct IBGP session or through VPNv6 route reflectors. The next hop of the advertising provider edge router still remains the IPv4 address (normally it is a loopback interface), but with the addition of IPv6, a value of ::FFFF: is prepended to the IPv4 next hop.

The technique can be best described as automatic tunneling of the IPv6 packets through the IPv4 backbone. The MP-BGP relationships remain the same as they are for VPNv4 traffic, with an additional capability of VPNv6. Where both IPv4 and IPv6 are supported, the same set of MPBGP peering relationships is used.

To summarize, from the control plane perspective, the prefixes are signaled across the backbone in the same way as regular MPLS and VPN prefix advertisements. The top label represents the IGP information that remains the same as for IPv4 MPLS. The bottom label represents the VPN information that the packet belongs to. As described earlier, additionally the MPBGP next hop is updated to make it IPv6-compliant. The forwarding or data plane function remains the same as it is deployed for the IPv4 MPLS VPN. The packet forwarding of IPv4 on the current MPLS VPN remains intact.

For detailed information on commands used to configure 6VPE over MPLS, see *Cisco IOS XR MPLS Configuration Guide*.

How to Implement MPLS Layer 3 VPNs

This section contains instructions for the following tasks:

- Configuring the Core Network, page VPC-285
- Connecting MPLS VPN Customers, page VPC-288
- Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels, page VPC-309 (optional)
- Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses, page VPC-318 (optional)
- Configuring Carrier Supporting Carrier, page VPC-329 (optional)
- Verifying the MPLS Layer 3 VPN Configuration, page VPC-337

Configuring the Core Network

Configuring the core network includes the following tasks:

- Assessing the Needs of MPLS VPN Customers, page VPC-285
- Configuring Routing Protocols in the Core, page VPC-286
- Configuring MPLS in the Core, page VPC-286
- Determining if FIB Is Enabled in the Core, page VPC-286
- Configuring Multiprotocol BGP on the PE Routers and Route Reflectors, page VPC-287

Assessing the Needs of MPLS VPN Customers

Before configuring an MPLS VPN, the core network topology must be identified so that it can best serve MPLS VPN customers. Perform this task to identify the core network topology.

SUMMARY STEPS

1. Identify the size of the network.
2. Identify the routing protocols in the core.
3. Determine if MPLS High Availability support is required.
4. Determine if BGP load sharing and redundant paths are required.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Identify the size of the network.	Identify the following to determine the number of routers and ports required: • How many customers will be supported? • How many VPNs are required for each customer? • How many virtual routing and forwarding (VRF) instances are there for each VPN?
Step 2	Identify the routing protocols in the core.	Determine which routing protocols are required in the core network.
Step 3	Determine if MPLS High Availability support is required.	MPLS VPN nonstop forwarding and graceful restart are supported on select routers and Cisco IOS XR software releases.
Step 4	Determine if BGP load sharing and redundant paths are required.	Determine if BGP load sharing and redundant paths in the MPLS VPN core are required.

Configuring Routing Protocols in the Core

To configure a routing protocol, see the *Cisco IOS XR Routing Configuration Guide*.

Configuring MPLS in the Core

To enable MPLS on all routers in the core, you must configure a Label Distribution Protocol (LDP). You can use either of the following as an LDP:

- MPLS LDP—See the *Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software* for configuration information.
- MPLS Traffic Engineering Resource Reservation Protocol (RSVP)—See *Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software* for configuration information.

Determining if FIB Is Enabled in the Core

Forwarding Information Base (FIB) must be enabled on all routers in the core, including the provider edge (PE) routers. For information on how to determine if FIB is enabled, see the *Implementing Cisco Express Forwarding on Cisco IOS XR Software* module in the *Cisco IOS XR IP Addresses and Services Configuration Guide*.

Configuring Multiprotocol BGP on the PE Routers and Route Reflectors

Perform this task to configure multiprotocol BGP (MP-BGP) connectivity on the PE routers and route reflectors.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family vpnv4 unicast**
   or
   address-family vpnv6 unicast
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family vpnv4 unicast**
   or
   address-family vpnv6 unicast
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`address-family vpnv4 unicast`**
	`or`
	`address-family vpnv6 unicast`
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
		Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 4	**`neighbor`** *`ip-address`* **`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp)# neighbor 172.168.40.24 remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number.

	Command or Action	Purpose

Step 5	**`address-family vpnv4 unicast`**
	`or`
	`address-family vpnv6 unicast`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
		Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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.

Connecting MPLS VPN Customers

To connect MPLS VPN customers to the VPN, perform the following tasks:

- Defining VRFs on the PE Routers to Enable Customer Connectivity, page VPC-289
- Configuring VRF Interfaces on PE Routers for Each VPN Customer, page VPC-291
- Configuring BGP as the Routing Protocol Between the PE and CE Routers, page VPC-293 (optional)
- Configuring RIPv2 as the Routing Protocol Between the PE and CE Routers, page VPC-297 (optional)
- Configuring Static Routes Between the PE and CE Routers, page VPC-300 (optional)
- Configuring OSPF as the Routing Protocol Between the PE and CE Routers, page VPC-301 (optional)
- Configuring EIGRP as the Routing Protocol Between the PE and CE Routers, page VPC-304 (optional)
- Configuring EIGRP Redistribution in the MPLS VPN, page VPC-307 (optional)

Defining VRFs on the PE Routers to Enable Customer Connectivity

Perform this task to define VPN routing and forwarding (VRF) instances.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-policy** *policy-name*
5. **import route-target** [*as-number:nn* | *ip-address:nn*]
6. **export route-policy** *policy-name*
7. **export route-target** [*as-number:nn* | *ip-address:nn*]
8. **exit**
9. **exit**
10. **router bgp** *autonomous-system-number*
11. **vrf** *vrf-name*
12. **rd** {*as-number* | *ip-address* | **auto**}
13. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config)# vrf vrf_1`	Configures a VRF instance and enters VRF configuration mode.
Step 3	**address-family ipv4 unicast** **Example:** `RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast`	Enters VRF address family configuration mode for the IPv4 address family.

Step 4	**import route-policy** *policy-name*
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# import route-policy policy_A`
		Specifies a route policy that can be imported into the local VPN.

	Command or Action	Purpose
Step 5	**`import route-target`** [*`as-number:nn`* | *`ip-address:nn`*] **Example:** `RP/0/0/CPU0:router(config-vrf-af)# import route-target 120:1`	Allows exported VPN routes to be imported into the VPN if one of the route targets of the exported route matches one of the local VPN import route targets.

Step 6	**`export route-policy`** *`policy-name`*
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# export route-policy policy_B`
		Specifies a route policy that can be exported from the local VPN.

Step 7	**`export route-target`** [*`as-number:nn`* | *`ip-address:nn`*] **Example:** `RP/0/0/CPU0:router(config-vrf-af)# export route-target 120:2`	Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 8	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# exit`
		Exits VRF address family configuration mode and returns the router to VRF configuration mode.

Step 9	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-vrf)# exit`
		Exits VRF configuration mode and returns the router to global configuration mode.

Step 10	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 11	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config-bgp)# vrf vrf_1`
		Configures a VRF instance and enters VRF configuration mode for BGP routing.

	Command or Action	Purpose

Step 12	**rd** {*as-number* | *ip-address* | **auto**}
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# rd auto`
		Automatically assigns a unique route distinguisher (RD) to vrf_1.

Step 13	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-vrf)# 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 VRF Interfaces on PE Routers for Each VPN Customer

Perform this task to associate a VPN routing and forwarding (VRF) instance with an interface or a subinterface on the PE routers.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address mask*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config)# interface GigabitEthernet 0/3/0/0`
		Enters interface configuration mode.

Step 3	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config-if)# vrf vrf_A`
		Configures a VRF instance and enters VRF configuration mode.

Step 4	**`ipv4 address`** *`ipv4-address mask`*
	Example:
	`RP/0/0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0`
		Configures a primary IPv4 address for the specified interface.

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-if)# end`
	or
	`RP/0/0/CPU0:router(config-if)# 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 BGP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure PE-to-CE routing sessions using BGP.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp router-id** {*ip-address*}
4. **vrf** *vrf-name*
5. **label-allocation-mode** *per-ce*
6. **address-family ipv4 unicast**
7. **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute isis *process-id* [**level** {**1** | **1-inter-area** | **2**}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute ospfv3 *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   or
   redistribute static [**metric** *metric-value*] [**route-policy** *route-policy-name*]
8. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
9. **network** {*ip-address/prefix-length* | *ip-address mask*} [**route-policy** *route-policy-name*]
10. **exit**
11. **neighbor** *ip-address*
12. **remote-as** *autonomous-system-number*
13. **password** {**clear** | **encrypted**} *password*
14. **ebgp-multihop** [*ttl-value*]
15. **address-family ipv4 unicast**
16. **allowas-in** [*as-occurrence-number*]
17. **route-policy** *route-policy-name* **in**
18. **route-policy** *route-policy-name* **out**
19. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config)# router bgp 120`	Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**bgp router-id** {*ip-address*}
	Example:
	`RP/0/0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24`
		Configures the local router with a router ID of 192.168.70.24.

Step 4	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-bgp)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for BGP routing.
Step 5	**label-allocation-mode per-ce** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce`	Sets the MPLS VPN label allocation mode for each customer edge (CE) label mode allowing the provider edge (PE) router to allocate one label for every immediate next-hop.
Step 6	**address-family ipv4 unicast** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast`	Enters VRF address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 7	**redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute isis** *process-id* [**level** {**1** | **1-inter-area** | **2**}] [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** route-policy-name] or **redistribute ospfv3** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**route-policy** *route-policy-name*] or **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*] **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-af)# redistribute connected`	Causes routes to be redistributed into BGP. The routes that can be redistributed into BGP are: • Connected • Intermediate System-to-Intermediate System (IS-IS) • Open Shortest Path First (OSPF) • Static
Step 8	**aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*] **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/8 as-set`	Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized. • The **as-set** keyword generates autonomous system set path information and community information from contributing paths. • The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths. • The **summary-only** keyword filters all more specific routes from updates. • The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.
Step 9	**network** {*ip-address/prefix-length* | *ip-address mask*} [**route-policy** *route-policy-name*] **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-af)# network 172.20.0.0/16`	Configures the local router to originate and advertise the specified network.
Step 10	**exit** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-af)# exit`	Exits VRF address family configuration mode and returns the router to VRF configuration mode for BGP routing.

	Command or Action	Purpose
Step 11	**neighbor** *ip-address* **Example:** `RP/0/0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24`	Places the router in VRF neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.
Step 12	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 13	**password** {**clear** | **encrypted**} *password* **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr)# password clear pswd123`	Configures neighbor 172.168.40.24 to use MD5 authentication with the password pswd123.
Step 14	**ebgp-multihop** [*ttl-value*] **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop`	Allows a BGP connection to neighbor 172.168.40.24.
Step 15	**address-family ipv4 unicast** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast`	Enters VRF neighbor address family configuration mode for BGP routing.
Step 16	**allowas-in** [*as-occurrence-number*] **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 3`	Replaces the neighbor autonomous system number (ASN) with the PE ASN in the AS path three times.
Step 17	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in`	Applies the In-Ipv4 policy to inbound IPv4 unicast routes.

	Command or Action	Purpose
Step 18	**route-policy** *route-policy-name* **out** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in`	Applies the In-Ipv4 policy to outbound IPv4 unicast routes.

Step 19	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# 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 RIPv2 as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions using Routing Information Protocol version 2 (RIPv2).

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **vrf** *vrf-name*
4. **interface** *type instance*
5. **site-of-origin** {*as-number*:*number* | *ip-address*:*number*}
6. **exit**
7. **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
   or
   redistribute connected [**route-policy** *name*]
   or
   redistribute isis *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
   or
   redistribute eigrp *as-number* [**route-policy** *name*]
   or

redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**route-policy** *name*]
or
redistribute static [**route-policy** *name*]

8. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router rip** **Example:** `RP/0/0/CPU0:router(config)# router rip`	Enters the Routing Information Protocol (RIP) configuration mode allowing you to configure the RIP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-rip)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for RIP routing.
Step 4	**interface** *type instance* **Example:** `RP/0/0/CPU0:router(config-rip-vrf)# interface GigabitEthernet 0/3/0/0`	Enters VRF interface configuration mode.

Step 5	**site-of-origin** {*as-number:number* | *ip-address:number*}
	Example:
	`RP/0/0/CPU0:router(config-rip-vrf-if)# site-of-origin 200:1`
		Identifies routes that have originated from a site so that the re-advertisement of that prefix back to the source site can be prevented. Uniquely identifies the site from which a PE router has learned a route.

Step 6	**exit** **Example:** `RP/0/0/CPU0:router(config-rip-vrf-if)# exit`	Exits VRF interface configuration mode, and returns the router to VRF configuration mode for RIP routing.

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td><b>Step 7</b></td><td><b>redistribute bgp</b> <i>as-number</i> [[<b>external</b> | <b>internal</b> | <b>local</b>] [<b>route-policy</b> <i>name</i>]
or
<b>redistribute connected</b> [<b>route-policy</b> <i>name</i>]
or
<b>redistribute eigrp</b> <i>as-number</i> [<b>route-policy</b> <i>name</i>]
or
<b>redistribute isis</b> <i>process-id</i> [<b>level-1</b> | <b>level-1-2</b> | <b>level-2</b>] [<b>route-policy</b> <i>name</i>]
or
<b>redistribute ospf</b> <i>process-id</i> [<b>match</b> {<b>external</b> [<b>1</b> | <b>2</b>] | <b>internal</b> | <b>nssa-external</b> [<b>1</b> | <b>2</b>]}] [<b>route-policy</b> <i>name</i>]
or
<b>redistribute static</b> [<b>route-policy</b> <i>name</i>]

<b>Example:</b>
RP/0/0/CPU0:router(config-rip-vrf)# redistribute connected</td><td>Causes routes to be redistributed into RIP. The routes that can be redistributed into RIP are:
• Border Gateway Protocol (BGP)
• Connected
• Enhanced Interior Gateway Routing Protocol (EIGRP)
• Intermediate System-to-Intermediate System (IS-IS)
• Open Shortest Path First (OSPF)
• Static</td></tr>
<tr><td><b>Step 8</b></td><td><b>end</b>
or
<b>commit</b>

<b>Example:</b>
RP/0/0/CPU0:router(config-rip-vrf)# end
or
RP/0/0/CPU0:router(config-rip-vrf)# commit</td><td>Saves configuration changes.
• When you issue the <b>end</b> command, the system prompts you to commit changes:
Uncommitted changes found, commit them before exiting(yes/no/cancel)?
[cancel]:
– Entering <b>yes</b> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.
– Entering <b>no</b> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.
– Entering <b>cancel</b> leaves the router in the current configuration session without exiting or committing the configuration changes.
• Use the <b>commit</b> command to save the configuration changes to the running configuration file and remain within the configuration session.</td></tr>
</table>

Configuring Static Routes Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use static routes.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf** *vrf-name*
4. **address-family ipv4 unicast**
5. *prefix/mask* [**vrf** *vrf-name*] {*ip-address* | *type interface-path-id*}
6. *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**
7. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router static** **Example:** `RP/0/0/CPU0:router(config)# router static`	Enters static routing configuration mode allowing you to configure the static routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-static)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for static routing.
Step 4	**address-family ipv4 unicast** **Example:** `RP/0/0/CPU0:router(config-static-vrf)# address-family ipv4 unicast`	Enters VRF address family configuration mode for the IPv4 address family.

	Command or Action	Purpose

Step 5	*prefix/mask* [**vrf** *vrf-name*] {*ip-address* | type interface-path-id}
	Example:
	`RP/0/0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 10.1.1.1`
		Assigns the static route to vrf_1.

Step 6	*prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect** **Example:** `RP/0/0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 bfd fast-detect`	Enables bidirectional forwarding detection (BFD) to detect failures in the path between adjacent forwarding engines. This option is available is when the forwarding router address is specified in Step 5.

Step 7	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-static-vrf-afi)# end`
	or
	`RP/0/0/CPU0:router(config-static-vrf-afi)# 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 OSPF as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Open Shortest Path First (OSPF).

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **router-id** {*router-id* | *type interface-path-id*}
5. **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   or
   redistribute connected [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

or
redistribute ospf *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
or
redistribute static [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
or
redistribute eigrp *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
or
redistribute rip [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

6. **area** *area-id*
7. **interface** *type interface-path-id*
8. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router ospf`** *`process-name`*
	Example:
	`RP/0/0/CPU0:router(config)# router ospf 109`
		Enters OSPF configuration mode allowing you to configure the OSPF routing process.

Step 3	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config-ospf)# vrf vrf_1`
		Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4	**`router-id`** `{`*`router-id`* `| type interface-path-id}`
	Example:
	`RP/0/0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10`
		Configures the router ID for the OSPF routing process.

	Command or Action	Purpose
Step 5	**redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute eigrp** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]}[**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] or **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*] **Example:** `RP/0/0/CPU0:router(config-ospf-vrf)# redistribute connected`	Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are: • Border Gateway Protocol (BGP) • Connected • Enhanced Interior Gateway Routing Protocol (EIGRP) • OSPF • Static • Routing Information Protocol (RIP)
Step 6	**area** *area-id* **Example:** `RP/0/0/CPU0:router(config-ospf-vrf)# area 0`	Configures the OSPF area as area 0.

	Command or Action	Purpose

Step 7	`interface type interface-path-id`
	Example:
	`RP/0/0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/3/0/0`
		Associates interface GigabitEthernet 0/3/0/0 with area 0.

Step 8	`end`
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-ospf-vrf-ar-if)# end`
	or
	`RP/0/0/CPU0:router(config-ospf-vrf-ar-if)# 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 EIGRP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Enhanced Interior Gateway Routing Protocol (EIGRP).

Using EIGRP between the PE and CE routers allows you to transparently connect EIGRP customer networks through an MPLS-enable Border Gateway Protocol (BGP) core network so that EIGRP routes are redistributed through the VPN across the BGP network as internal BGP (iBGP) routes.

Prerequisites

BGP must configured in the network. See the *Implementing BGP on Cisco IOS XR Software* module in *Cisco IOS XR Routing Configuration Guide*.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **router-id** *router-id*
6. **autonomous-system** *as-number*
7. **default-metric** *bandwidth delay reliability loading mtu*
8. **redistribute** {{**bgp** | **connected** | **isis** | **ospf**| **rip** | **static**} [*as-number* | *instance-name*]} [**route-policy** *name*]
9. **interface** *type interface-path-id*
10. **site-of-origin** {*as-number:number* | *ip-address:number*}
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router eigrp** *as-number* **Example:** `RP/0/0/CPU0:router(config)# router eigrp 24`	Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-eigrp)# vrf vrf_1`	Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for EIGRP routing.
Step 4	**address-family ipv4** **Example:** `RP/0/0/CPU0:router(config-eigrp-vrf)# address family ipv4`	Enters VRF address family configuration mode for the IPv4 address family.
Step 5	**router-id** *router-id* **Example:** `RP/0/0/CPU0:router(config-eigrp-vrf-af)# router-id 172.20.0.0`	Configures the router ID for the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process.

	Command or Action	Purpose

Step 6	**`autonomous-system`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 6`
		Configures the EIGRP routing process to run within a VRF.

Step 7	**`default-metric`** *`bandwidth delay reliability loading mtu`*
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af)# default-metric 100000 4000 200 45 4470`
		Sets the metrics for an EIGRP.

Step 8	**`redistribute`** `{{`**`bgp`** `|` **`connected`** `|` **`isis`** `|` **`ospf`**`|` **`rip`** `|` **`static`**`}` `[`*`as-number`* `|` *`instance-name`*`]}` `[`**`route-policy`** *`name`*`]`
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af)# redistribute connected`
		Causes connected routes to be redistributed into EIGRP.

Step 9	**`interface`** `type interface-path-id`
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af)# interface GigabitEthernet 0/3/0/0`
		Associates interface GigabitEthernet 0/3/0/0 with the EIGRP routing process.

Step 10	**`site-of-origin`** `{`*`as-number:number`* `|` *`ip-address:number`*`}`
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 201:1`
		Configures site of origin (SoO) on interface GigabitEthernet 0/3/0/0.

Step 11	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af-if)# end`
	or
	`RP/0/0/CPU0:router(config-eigrp-vrf-af-if)# 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 EIGRP Redistribution in the MPLS VPN

Perform this task for every provider edge (PE) router that provides VPN services to enable Enhanced Interior Gateway Routing Protocol (EIGRP) redistribution in the MPLS VPN.

Prerequisites

The metric can be configured in the route-policy configuring using the **redistribute** command (or configured with the **default-metric** command). If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not installed in the EIGRP database. If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not advertised to the CE router. See the *Implementing EIGRP on Cisco IOS XR Software* module in the *Cisco IOS XR Routing Configuration Guide*.

Restrictions

Redistribution between native EIGRP VPN routing and forwarding (VRF) instances is not supported. This behavior is designed.

SUMMARY STEPS

1. **configure**
2. **router eigrp** as-number
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**router eigrp** *as-number* **Example:** `RP/0/0/CPU0:router(config)# router eigrp 24`	Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.
Step 3	**vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router(config-eigrp)# vrf vrf_1`	Configures a VRF instance and enters VRF configuration mode for EIGRP routing.

	Command or Action	Purpose

Step 4	**`address-family ipv4`**
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf)# address family ipv4`
		Enters VRF address family configuration mode for the IPv4 address family.

Step 5	**`redistribute bgp`** [*`as-number`*] [**`route-policy`** *`policy-name`*] **Example:** `RP/0/0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 24 route-policy policy_A`	Causes Border Gateway Protocol (BGP) routes to be redistributed into EIGRP.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-eigrp-vrf-af-if)# end`
	or
	`RP/0/0/CPU0:router(config-eigrp-vrf-af-if)# 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.

Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels

Note This section is not applicable to Inter-AS over IP tunnels.

This section contains instructions for the following tasks:

- Configuring ASBRs to Exchange IPv4 Routes and MPLS Labels, page VPC-310
- Configuring the Route Reflectors to Exchange VPN-IPv4 Routes, page VPC-312
- Configuring the Route Reflector to Reflect Remote Routes in its AS, page VPC-315

Configuring ASBRs to Exchange IPv4 Routes and MPLS Labels

Perform this task to configure the autonomous system boundary routers (ASBRs) to exchange IPv4 routes and MPLS labels.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family ipv4 unicast**
4. **allocate-label all**
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **address-family ipv4 labeled-unicast**
8. **route-policy** *route-policy-name* **in**
9. **route-policy** *route-policy-name* **out**
10. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**address-family ipv4 unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Enters global address family configuration mode for the IPv4 unicast address family.

Step 4	**allocate-label all** **Example:** `RP/0/0/CPU0:router(config-bgp-af)# allocate-label all`	Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

	Command or Action	Purpose

Step 5	**neighbor** *ip-address*
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 6	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

Step 7	**address-family ipv4 labeled-unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)`
		Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 8	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`	Applies a routing policy to updates that are received from a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **in** keyword to define the policy for inbound routes.

	Command or Action	Purpose

Step 9	**`route-policy`** *`route-policy-name`* **`out`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`
		Applies a routing policy to updates that are sent to a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **out** keyword to define the policy for outbound routes.

Step 10	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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 the Route Reflectors to Exchange VPN-IPv4 Routes

Perform this task to enable the route reflectors to exchange VPN-IPv4 routes by using multihop. This task specifies that the next-hop information and the VPN label are to be preserved across the autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **ebgp-multihop** [*ttl-value*]
6. **update-source** *type interface-path-id*
7. **address-family vpnv4 unicast**
8. **route-policy** *route-policy-name* **in**
9. **route-policy** *route-policy-name* **out**

10. **next-hop-unchanged**
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/0/CPU0:router(config-bgp)# neighbor 172.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number.

Step 5	**`ebgp-multihop`** `[`*`ttl-value`*`]`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop`
		Enables multihop peerings with external BGP neighbors.

Step 6	**`update-source`** *`type interface-path-id`*
	Example:
	`RP/0/0/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 7	**`address-family vpnv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

	Command or Action	Purpose
Step 8	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`	Applies a routing policy to updates that are received from a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **in** keyword to define the policy for inbound routes.
Step 9	**route-policy** *route-policy-name* **out** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`	Applies a routing policy to updates that are sent to a BGP neighbor. • Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all. • Use the **out** keyword to define the policy for outbound routes.
Step 10	**next-hop-unchanged** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# next-hop-unchanged`	Disables overwriting of the next hop before advertising to external Border Gateway Protocol (eBGP) peers.

Step 11	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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 the Route Reflector to Reflect Remote Routes in its AS

Perform this task to enable the route reflector (RR) to reflect the IPv4 routes and labels learned by the autonomous system boundary router (ASBR) to the provider edge (PE) routers in the autonomous system. This task is accomplished by making the ASBR and PE route reflector clients of the RR.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family ipv4 unicast**
4. **allocate-label all**
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **update-source** *type interface-path-id*
8. **address-family ipv4 labeled-unicast**
9. **route-reflector-client**
10. **neighbor** *ip-address*
11. **remote-as** *autonomous-system-number*
12. **update-source** *type interface-path-id*
13. **address-family ipv4 labeled-unicast**
14. **route-reflector-client**
15. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Enters global address family configuration mode for the IPv4 unicast address family.

	Command or Action	Purpose
Step 4	**allocate-label all** **Example:** `RP/0/0/CPU0:router(config-bgp-af)# allocate-label all`	Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5	**neighbor** *ip-address*
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.
Step 7	**update-source** *type interface-path-id* **Example:** `RP/0/0/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 8	**address-family ipv4 labeled-unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 9	**route-reflector-client** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`	Configures the router as a BGP route reflector and neighbor 172.168.40.24 as its client.

Step 10	**neighbor** *ip-address*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# neighbor 10.40.25.2`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address .40.25.2 as an VPNv4 iBGP peer.

Step 11	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

	Command or Action	Purpose

Step 12	**`update-source`** *`type interface-path-id`*
	Example:
	`RP/0/0/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 13	**`address-family ipv4 labeled-unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 14	**`route-reflector-client`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client`
		Configures the neighbor as a route reflector client.

Step 15	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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.

Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses

This section contains instructions for the following tasks:

- Configuring the ASBRs to Exchange VPN-IPv4 Addresses, page VPC-318
- Configuring a Static Route to an ASBR Peer, page VPC-321
- Configuring EBGP Routing to Exchange VPN Routes Between Subautonomous Systems in a Confederation, page VPC-323
- Configuring MPLS Forwarding for ASBR Confederations, page VPC-325
- Configuring a Static Route to an ASBR Confederation Peer, page VPC-327

Configuring the ASBRs to Exchange VPN-IPv4 Addresses

Perform this task to configure an external Border Gateway Protocol (eBGP) autonomous system boundary router (ASBR) to exchange VPN-IPv4 routes with another autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family vpnv4 unicast**
4. **neighbor** *ip-address*
5. **remote-as** *autonomous-system-number*
6. **address-family vpnv4 unicast**
7. **route-policy** *route-policy-name* **in**
8. **route-policy** *route-policy-name* **out**
9. **neighbor** *ip-address*
10. **remote-as** *autonomous-system-number*
11. **update-source** *type interface-path-id*
12. **address-family vpnv4 unicast**
13. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3	**`address-family vpnv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

Step 4	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 5	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number.

Step 6	**`address-family vpnv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 7	**`route-policy`** *`route-policy-name`* **`in`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in`
		Applies a routing policy to updates that are received from a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **in** keyword to define the policy for inbound routes.

	Command or Action	Purpose

Step 8	**`route-policy`** *`route-policy-name`* **`out`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out`
		Applies a routing policy to updates that are sent from a BGP neighbor.
		• Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
		• Use the **out** keyword to define the policy for outbound routes.

Step 9	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# neighbor 10.40.25.2`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 10.40.25.2 as an VPNv4 iBGP peer.

Step 10	**`remote-as`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`
		Creates a neighbor and assigns it a remote autonomous system number.

Step 11	**`update-source`** *`type interface-path-id`*
	Example:
	`RP/0/0/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.

	Command or Action	Purpose

Step 12	**`address-family vpnv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 13	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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 a Static Route to an ASBR Peer

Perform this task to configure a static route to an ASBR peer.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router static`**
	Example:
	`RP/0/0/CPU0:router(config)# router static`
	`RP/0/0/CPU0:router(config-static)#`
		Enters router static configuration mode.

Step 3	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-static)#`
	`address-family ipv4 unicast`
	`RP/0/0/CPU0:router(config-static-afi)#`
		Enables an IPv4 address family.

Step 4	***`A.B.C.D/length`*** *`next-hop`*
	Example:
	`RP/0/0/CPU0:router(config-static-afi)#`
	`10.10.10.10/32 10.9.9.9`
		Enters the address of the destination router (including IPv4 subnet mask).

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-static-afi)# end`
	or
	`RP/0/0/CPU0:router(config-static-afi)# 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 EBGP Routing to Exchange VPN Routes Between Subautonomous Systems in a Confederation

Perform this task to configure external Border Gateway Protocol (eBGP) routing to exchange VPN routes between subautonomous systems in a confederation.

Note To ensure that host routes for VPN-IPv4 eBGP neighbors are propagated (by means of the Interior Gateway Protocol [IGP]) to other routers and PE routers, specify the **redistribute connected** command in the IGP configuration portion of the confederation eBGP (CEBGP) router. If you are using Open Shortest Path First (OSPF), make sure that the OSPF process is not enabled on the CEBGP interface in which the “redistribute connected” subnet exists.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp confederation peers** *peer autonomous-system-number*
4. **bgp confederation identifier** *autonomous-system-number*
5. **address-family vpnv4 unicast**
6. **neighbor** *ip-address*
7. **remote-as** *autonomous-system-number*
8. **address-family vpnv4 unicast**
9. **route-policy** *route-policy-name* **in**
10. **route-policy** *route-policy-name* **out**
11. **next-hop-self**
12. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`autonomous-system-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

	Command or Action	Purpose
Step 3	**bgp confederation peers** *peer autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp)# bgp confederation peers 8`	Configures the peer autonomous system number that belongs to the confederation.
Step 4	**bgp confederation identifier** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp)# bgp confederation identifier 5`	Specifies the autonomous system number for the confederation ID.

Step 5	**address-family vpnv4 unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

Step 6	**neighbor** *ip-address*
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.168.40.24`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 10.168.40.24 as a BGP peer.

Step 7	**remote-as** *autonomous-system-number* **Example:** `RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 2002`	Creates a neighbor and assigns it a remote autonomous system number.

Step 8	**address-family vpnv4 unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 address family.

Step 9	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in`	Applies a routing policy to updates received from a BGP neighbor.
Step 10	**route-policy** *route-policy-name* **out** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy Out-Ipv4 out`	Applies a routing policy to updates advertised to a BGP neighbor.

	Command or Action	Purpose
Step 11	**next-hop-self** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# next-hop-self`	Disables next-hop calculation and let you insert your own address in the next-hop field of BGP updates.

Step 12	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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 MPLS Forwarding for ASBR Confederations

Perform this task to configure MPLS forwarding for autonomous system boundary router (ASBR) confederations (in BGP) on a specified interface.

Note This configuration adds the implicit NULL rewrite corresponding to the peer associated with the interface, which is required to prevent BGP from automatically installing rewrites by LDP (in multihop instances).

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **mpls activate**
4. **interface** *type interface-path-id*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)`
		Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3	**`mpls activate`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# mpls activate`
	`RP/0/0/CPU0:router(config-bgp-mpls)#`
		Enters BGP MPLS activate configuration mode.

Step 4	**`interface`** *`type interface-path-id`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-mpls)# interface GigabitEthernet 0/3/0/0`
		Enables MPLS on the interface.

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-bgp-mpls)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-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.

Configuring a Static Route to an ASBR Confederation Peer

Perform this task to configure a static route to an Inter-AS confederation peer. For more detailed information, see "Configuring a Static Route to a Peer" section on page MPC-335.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router static`**
	Example:
	`RP/0/0/CPU0:router(config)# router static`
	`RP/0/0/CPU0:router(config-static)#`
		Enters router static configuration mode.

Step 3	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-static)#`
	`address-family ipv4 unicast`
	`RP/0/0/CPU0:router(config-static-afi)#`
		Enables an IPv4 address family.

	Command or Action	Purpose
Step 4	***A.B.C.D/length*** *next-hop* **Example:** `RP/0/0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9`	Enters the address of the destination router (including IPv4 subnet mask).

Step 5	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-static-afi)# end`
	or
	`RP/0/0/CPU0:router(config-static-afi)# 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 Carrier Supporting Carrier

Perform the tasks in this section to configure Carrier Supporting Carrier (CSC):

- Identifying the Carrier Supporting Carrier Topology, page VPC-329
- Configuring the Backbone Carrier Core, page VPC-330
- Configuring the CSC-PE and CSC-CE Routers, page VPC-330
- Configuring a Static Route to a Peer, page VPC-335

Identifying the Carrier Supporting Carrier Topology

Before you configure the MPLS VPN CSC with BGP, you must identify both the backbone and customer carrier topology.

Note You can connect multiple CSC-CE routers to the same PE, or you can connect a single CSC-CE router to multiple CSC-PEs using more than one CSC-CE interface to provide redundancy and multiple path support in a CSC topology.

Perform this task to identify the carrier supporting carrier topology.

SUMMARY STEPS

1. Identify the type of customer carrier, ISP, or MPLS VPN service provider.
2. Identify the CE routers.
3. Identify the customer carrier core router configuration.
4. Identify the customer carrier edge (CSC-CE) routers.
5. Identify the backbone carrier router configuration.

DETAILED STEPS

	Command or Action	Purpose
Step 1	Identify the type of customer carrier, ISP, or MPLS VPN service provider.	Sets up requirements for configuration of carrier supporting carrier network.
Step 2	Identify the CE routers.	Sets up requirements for configuration of CE to PE connections.
Step 3	Identify the customer carrier core router configuration.	Sets up requirements for configuration between core (P) routers and between P routers and edge routers (PE and CSC-CE routers).
Step 4	Identify the customer carrier edge (CSC-CE) routers.	Sets up requirements for configuration of CSC-CE to CSC-PE connections.
Step 5	Identify the backbone carrier router configuration.	Sets up requirements for configuration between CSC core routers and between CSC core routers and edge routers (CSC-CE and CSC-PE routers).

Configuring the Backbone Carrier Core

Configuring the backbone carrier core requires setting up connectivity and routing functions for the CSC core and the CSC-PE routers. To do so, you must complete the following high-level tasks:

- Verify IP connectivity in the CSC core.
- Verify LDP configuration in the CSC core.

Note This task is not applicable to CSC over IP tunnels.

- Configure VRFs for CSC-PE routers.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

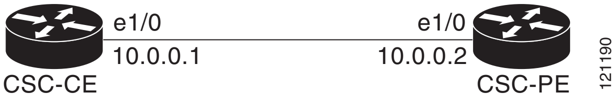
Configuring the CSC-PE and CSC-CE Routers

Perform the following tasks to configure links between a CSC-PE router and the carrier CSC-CE router for an MPLS VPN CSC network that uses BGP to distribute routes and MPLS labels:

- Configuring a CSC-PE
- Configuring a CSC-CE

Figure 37 shows the configuration for the peering with directly connected interfaces between CSC-PE and CSC-CE routers. This configuration is used as the example in the tasks that follow.

Figure 37 ***Configuration for Peering with Directly Connected Interfaces Between CSC-PE and CSC-CE Routers***



Configuring a CSC-PE

Perform this task to configure a CSC-PE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family vpnv4 unicast**
4. **neighbor** *A.B.C.D*
5. **remote-as** *as-number*
6. **update-source** *type interface-path-id*
7. **address-family vpnv4 unicast**
8. **vrf** *vrf-name*
9. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}

10. **address-family ipv4 unicast**
11. **allocate-label all**
12. **neighbor** *A.B.C.D*
13. **remote-as** *as-number*
14. **address-family ipv4 labeled-unicast**
15. **route-policy** *route-policy-name* **in**
16. **route-policy** *route-policy-name* **out**
17. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 2`
	`RP/0/0/CPU0:router(config-bgp)#`
		Configures a BGP routing process and enters router configuration mode.
		• Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3	**`address-family vpnv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)#`
		Configures VPNv4 address family.

Step 4	**`neighbor`** *`A.B.C.D`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.10.10.0`
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
		Configures the IP address for the BGP neighbor.

Step 5	**`remote-as`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 888`
		Configures the AS number for the BGP neighbor.

Step 6	**`update-source`** *`type interface-path-id`*
	Example:
	`RP/0/0/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.

	Command or Action	Purpose

Step 7	**`address-family vpnv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
	`address-family vpnv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures VPNv4 unicast address family.

Step 8	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf 9999`
	`RP/0/0/CPU0:router(config-bgp-vrf)#`
		Configures a VRF instance.

Step 9	**`rd`** `{`*`as-number:nn`* `|` *`ip-address:nn`* `|` **`auto`**`}`
	Example:
	`RP/0/0/CPU0:router(onfig-bgp-vrf)# rd auto`
		Configures a route distinguisher.
		Note Use the **auto** keyword to automatically assign a unique route distinguisher.

Step 10	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)#`
	`address-family ipv4 unicast`
	`RP/0/0/CPU0:router(config-bgp-vrf-af)#`
		Configures IPv4 unicast address family.

Step 11	**`allocate-label all`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-af)#`
	`allocate-label all`
		Allocate labels for all local prefixes and prefixes received with labels.

Step 12	**`neighbor`** *`A.B.C.D`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-af)# neighbor`
	`10.10.10.0`
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)#`
		Configures the IP address for the BGP neighbor.

Step 13	**`remote-as`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)#`
	`remote-as 888`
		Enables the exchange of information with a neighboring BGP router.

Step 14	**`address-family ipv4 labeled-unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr)#`
	`address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)#`
		Configures IPv4 labeled-unicast address family.

	Command or Action	Purpose

Step 15	**`route-policy`** *`route-policy-name`* **`in`**
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all in`
		Applies the pass-all policy to all inbound routes.

Step 16	**`route-policy`** *`route-policy-name`* **`out`**
	
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out`
		Applies the pass-all policy to all outbound routes.

Step 17	**`end`**
	or
	`commit`
	
	Example:
	`RP/0/0/CPU0:router(cconfig-bgp-vrf-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# 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 a CSC-CE

Perform this task to configure a CSC-CE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv4 unicast**
4. **redistribute ospf** *instance-number*
5. **allocate-label route-policy** *route-policy-name*
6. **exit**
7. **neighbor** *A.B.C.D*
8. **remote-as** *as-number*
9. **address-family ipv4 labeled-unicast**

10. **route-policy** *route-policy-name* **in**
11. **route-policy** *route-policy-name* **out**
12. **end**
or
commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`router bgp`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 1`
		Configures a BGP routing process and enters router configuration mode.
		• Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast`
		Configures IPv4 unicast address-family.

Step 4	**`redistribute ospf`** *`instance-number`*
	Example:
	`RP/0/0/CPU0:router(config-router-af)# redistribute ospf 1`
		Redistributes OSPF routes into BGP.

Step 5	**`allocate-label route-policy`** *`route-policy-name`*
	Example:
	`RP/0/0/CPU0:router(config-router-af)# allocate-label route-policy internal-routes`
		Allocates labels for those routes that match the route policy. These labeled routes are advertised to neighbors configured with address-family ipv4 labeled-unicast.

Step 6	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-af)# exit`
		Exits the current configuration mode.

Step 7	**`neighbor`** *`A.B.C.D`*
	Example:
	`RP/0/0/CPU0:router(config-bgp)# neighbor 10.0.0.1`
		Configures the IP address for the BGP neighbor.

Step 8	**`remote-as`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 1`
		Enables the exchange of information with a neighboring BGP router.

	Command or Action	Purpose

Step 9	**address-family ipv4 labeled-unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)#`
	`address-family ipv4 labeled-unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
		Configures IPv4 labeled-unicast address family.

Step 10	**route-policy** *route-policy-name* **in**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
	`route-policy pass-all in`
		Applies the route-policy to all inbound routes.

Step 11	**route-policy** *route-policy-name* **out**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)#`
	`route-policy pass-all out`
		Applies the route-policy to all outbound routes.

Step 12	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp)#` **end**
	or
	`RP/0/0/CPU0:router(config-bgp)#` **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 a Static Route to a Peer

Perform this task to configure a static route to an Inter-AS or CSC-CE peer.

When you configure an Inter-AS or CSC peer, BGP allocates a label for a /32 route to that peer and performs a NULL label rewrite. When forwarding a labeled packet to the peer, the router removes the top label from the label stack; however, in such an instance, BGP expects a /32 route to the peer. This task ensures that there is, in fact, a /32 route to the peer.

Please be aware of the following facts before performing this task:

- A /32 route is not required to establish BGP peering. A route using a shorter prefix length will also work.
- A shorter prefix length route is not associated with the allocated label; even though the BGP session comes up between the peers, without the static route, forwarding will not work.

Note To configure a static route on a CSC-PE, you must configure the router under the VRF (as noted in the detailed steps).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router(config)# configure`
		Enters global configuration mode.

Step 2	**`router static`**
	Example:
	`RP/0/0/CPU0:router(config)# router static`
		Enters router static configuration mode.

Step 3	**`address-family ipv4 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-static)# address-family ipv4 unicast`
		Enables an IPv4 address family.
		Note To configure a static route on a CSC-PE, you must first configure the VRF using the **vrf** command before **address-family**.

	Command or Action	Purpose
Step 4	***A.B.C.D/length*** *next-hop* **Example:** `RP/0/0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9`	Enters the address of the destination router (including IPv4 subnet mask).

Step 5	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-static-af)#` **`end`**
	or
	`RP/0/0/CPU0:router(config-static-af)#` **`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.

Verifying the MPLS Layer 3 VPN Configuration

Perform this task to verify the MPLS Layer 3 VPN configuration.

SUMMARY STEPS

1. **show running-config router bgp** *as-number* **vrf** *vrf-name*
2. **show running-config routes**
3. **show ospf vrf** *vrf-name* **database**
4. **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*
5. **show bgp vrf** *vrf-name* **summary**
6. **show bgp vrf** *vrf-name* **neighbors** *ip-address*
7. **show bgp vrf** *vrf-name*
8. **show route vrf** *vrf-name ip-address*
9. **show bgp vpn unicast summary**
10. **show running-config router isis**
11. **show running-config mpls**
12. **show isis adjacency**

13. **show mpls ldp forwarding**
14. **show bgp vpnv4 unicast**
    or
    show bgp vpnv6 unicast
15. **show bgp vrf** *vrf-name*
16. **show bgp vrf** *vrf-name* **imported-routes**
17. **show route vrf** *vrf-name ip-address*
18. **show cef vrf** *vrf-name ip-address*
19. **show cef vrf** *vrf-name ip-address* **location** *node-id*
20. **show bgp vrf** *vrf-name ip-address*
21. **show ospf vrf** *vrf-name* **database**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show running-config router bgp** *as-number* **vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router# show running-config router bgp 3 vrf vrf_A`	Displays the specified VPN routing and forwarding (VRF) content of the currently running configuration.
Step 2	**show running-config routes** **Example:** `RP/0/0/CPU0:router# show running-config routes`	Displays the Open Shortest Path First (OSPF) routes table in the currently running configuration.
Step 3	**show ospf vrf** *vrf-name* **database** **Example:** `RP/0/0/CPU0:router# show ospf vrf vrf_A database`	Displays lists of information related to the OSPF database for a specified VRF.
Step 4	**show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address* **Example:** `RP/0/0/CPU0:router# show running-config router bgp 3 vrf vrf_A neighbor 172.168.40.24`	Displays the Border Gateway Protocol (BGP) VRF neighbor content of the currently running configuration.
Step 5	**show bgp vrf** *vrf-name* **summary** **Example:** `RP/0/0/CPU0:router# show bgp vrf vrf_A summary`	Displays the status of the specified BGP VRF connections.
Step 6	**show bgp vrf** *vrf-name* **neighbors** *ip-address* **Example:** `RP/0/0/CPU0:router# show bgp vrf vrf_A neighbors 172.168.40.24`	Displays information about BGP VRF connections to the specified neighbors.

	Command or Action	Purpose
Step 7	**show bgp vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router# show bgp vrf vrf_A`	Displays information about a specified BGP VRF.
Step 8	**show route vrf** *vrf-name ip-address* **Example:** `RP/0/0/CPU0:router# show route vrf vrf_A 10.0.0.0`	Displays the current routes in the Routing Information Base (RIB) for a specified VRF.
Step 9	**show bgp vpn unicast summary** **Example:** `RP/0/0/CPU0:router# show bgp vpn unicast summary`	Displays the status of all BGP VPN unicast connections.
Step 10	**show running-config router isis** **Example:** `RP/0/0/CPU0:router# show running-config router isis`	Displays the Intermediate System-to-Intermediate System (IS-IS) content of the currently running configuration.
Step 11	**show running-config mpls** **Example:** `RP/0/0/CPU0:router# show running-config mpls`	Displays the MPLS content of the currently running-configuration.
Step 12	**show isis adjacency** **Example:** `RP/0/0/CPU0:router# show isis adjacency`	Displays IS-IS adjacency information.
Step 13	**show mpls ldp forwarding** **Example:** `RP/0/0/CPU0:router# show mpls ldp forwarding`	Displays the Label Distribution Protocol (LDP) forwarding state installed in MPLS forwarding.
Step 14	**show bgp vpnv4 unicast** or **show bgp vpnv6 unicast** **Example:** `RP/0/0/CPU0:router# show bgp vpnv4 unicast`	Displays entries in the BGP routing table for VPNv4 or VPNv6 unicast addresses.
Step 15	**show bgp vrf** *vrf-name* **Example:** `RP/0/0/CPU0:router# show bgp vrf vrf_A`	Displays entries in the BGP routing table for VRF vrf_A.
Step 16	**show bgp vrf** *vrf-name* **imported-routes** **Example:** `RP/0/0/CPU0:router# show bgp vrf vrf_A imported-routes`	Displays BGP information for routes imported into specified VRF instances.

	Command or Action	Purpose
Step 17	**show route vrf** *vrf-name ip-address* **Example:** `RP/0/0/CPU0:router# show route vrf vrf_A 10.0.0.0`	Displays the current specified VRF routes in the RIB.
Step 18	**show cef vrf** *vrf-name ip-address* **Example:** `RP/0/0/CPU0:router# show cef vrf vrf_A 10.0.0.1`	Displays the IPv4 Cisco Express Forwarding (CEF) table for a specified VRF.
Step 19	**show cef vrf** *vrf-name ip-address* **location** *node-id* **Example:** `RP/0/0/CPU0:router# show cef vrf vrf_A 10.0.0.1 location 0/1/cpu0`	Displays the IPv4 CEF table for a specified VRF and location.
Step 20	**show bgp vrf** *vrf-name ip-address* **Example:** `RP/0/0/CPU0:router# show bgp vrf vrf_A 10.0.0.0`	Displays entries in the BGP routing table for VRF vrf_A.
Step 21	**show ospf vrf** *vrf-name* **database** **Example:** `RP/0/0/CPU0:router# show ospf vrf vrf_A database`	Displays lists of information related to the OSPF database for a specified VRF.

Configuring 6VPE Support

The following tasks are required to configure 6VPE support:

- Configuring an IPv6 Address Family Under VRF, page VPC-341
- Configuring BGP Route Distinguisher and Core-facing Sessions, page VPC-342
- Configuring a PE-CE Protocol, page VPC-344

Configuring an IPv6 Address Family Under VRF

Perform this task to configure an IPv6 address-family under the VRF for 6VPE support.

Note You can also configure a maximum-routes limit for the VRF, export, and import policies.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf_name*
3. **address-family ipv6 unicast**
4. **import route-target** [*as-number:nn* | *ip-address:nn*]
5. **export route-target** [*as-number:nn* | *ip-address:nn*]
6. **end**
   or
   commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router# configure`
		Enters global configuration mode.

Step 2	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config)# vrf vrf_1`
		Configures a VRF instance and enters VRF configuration mode.

Step 3	**`address-family ipv6 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast`
		Enters VRF address family configuration mode for the IPv6 address family.

Step 4	**`import route-target`** `[`*`as-number:nn`* `|` *`ip-address:nn`*`]`
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# import route-target 120:1`
		Configures a VPN routing and forwarding (VRF) import route-target extended community.

	Command or Action	Purpose

Step 5	**`export route-target`** `[`*`as-number:nn`* `|`
	`ip-address:nn``]`
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# export route-target 120:2`
		Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 6	**`end`**
	or
	`commit`
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# end`
	or
	`RP/0/0/CPU0:router(config-vrf-af)# 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 BGP Route Distinguisher and Core-facing Sessions

Perform this task to configure VRF route distinguisher values and core-facing neighbors under BGP.

Note Before you perform this task, you must first configure a VRF and map the VRF to an interface. For more information, see *Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software*.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family vpnv6 unicast**
4. **vrf** *vrf-name*
5. **rd** {*as-number:nn* | *ip-address:nn* | **auto**}
6. **address-family ipv6 unicast**
7. **exit**
8. **neighbor** *ip-address* **remote-as** *as-number*

9. **address-family ipv6 unicast**
10. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router#` **`configure`**
		Enters global configuration mode.

Step 2	**`router bgp`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 100`
	`RP/0/0/CPU0:router(config-bgp)#`
		Enters router BGP configuration mode.

Step 3	**`address-family vpnv6 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp)# address-family vpnv6 unicast`
	`RP/0/0/CPU0:router(config-bgp-af)`
		Enters address family configuration mode for the VPNv6 address family.

Step 4	**`vrf`** *`vrf-name`*
	Example:
	`RP/0/0/CPU0:router(config-bgp)# vrf red`
		Configures a VPN VRF instance and enters VRF configuration mode.

Step 5	**`rd`** {*`as-number:nn`* | *`ip-address:nn`* | **`auto`**}
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# router bgp 100`
		Configures a route distinguisher.

Step 6	**`address-family ipv6 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast`
		Enters IPv6 address family configuration mode.

Step 7	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-af)# exit`
		Exits the current configuration mode.

Step 8	**`neighbor`** *`ip-address`* **`remote-as`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24 remote-as 2002f`
		Creates a neighbor and assigns it a remote autonomous system number.

	Command or Action	Purpose
Step 9	**address-family ipv6 unicast** **Example:** `RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast`	Enters IPv6 address family configuration mode.

Step 10	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-vrf-af)# end`
	or
	`RP/0/0/CPU0:router(config-vrf-af)# 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 a PE-CE Protocol

Perform this task to configure a PE-CE protocol for 6VPE.

Note eBGP, iBGP and eiBGP load-balancing configuration options are also supported for 6VPE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv6 unicast**
5. **exit**
6. **exit**
7. **neighbor** *ip-address*
8. **remote-as** *as-number*
9. **address-family vpnv6 unicast**

10. **route-policy** *route-policy-name* **in**
11. **end**
    or
    commit

DETAILED STEPS

	Command or Action	Purpose

Step 1	**`configure`**
	Example:
	`RP/0/0/CPU0:router#` **`configure`**
		Enters global configuration mode.

Step 2	**`router bgp`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config)# router bgp 120`
	`RP/0/0/CPU0:router(config-bgp)`
		Enters router BGP configuration mode.

Step 3	**`vrf`** *`vrf-name`*
	`Example:`
	`RP/0/0/CPU0:router(config-bgp)# vrf red`
	`RP/0/0/CPU0:router(config-bgp-vrf)`
		Configures a VPN VRF instance and enters VRF configuration mode.

Step 4	**`address-family ipv6 unicast`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf) address-family ipv6 unicast`
	`RP/0/0/CPU0:router(config-bgp-vrf-af)`
		Enters IPv6 address family configuration mode.

Step 5	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf-af)# exit`
		Exits the current configuration mode.

Step 6	**`exit`**
	Example:
	`RP/0/0/CPU0:router(config-bgp-vrf)# exit`
		Exits the current configuration mode.

Step 7	**`neighbor`** *`ip-address`*
	Example:
	`RP/0/0/CPU0:router(config-bgp)# neighbor 10,10.10,10`
		Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 8	**`remote-as`** *`as-number`*
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 1000`
		Creates a BGP neighbor and begin the exchange of routing information.

	Command or Action	Purpose

Step 9	**address-family vpnv6 unicast**
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast`
	`RP/0/0/CPU0:router(config-bgp-nbr-af)`
		Enters address family configuration mode for the VPNv6 address family.

Step 10	**route-policy** *route-policy-name* **in** **Example:** `RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in`	Applies a routing policy to updates advertised to or received from a BGP neighbor.

Step 11	**end**
	or
	commit
	Example:
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# end`
	or
	`RP/0/0/CPU0:router(config-bgp-nbr-af)# 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.

Configuration Examples for Implementing MPLS Layer 3 VPNs

The following section provides sample configurations for MPLS L3VPN features, including:

- Configuring an MPLS VPN Using BGP: Example, page VPC-347
- Configuring the Routing Information Protocol on the PE Router: Example, page VPC-348
- Configuring the PE Router Using EIGRP: Example, page VPC-348
- Configuration Examples for MPLS VPN CSC, page VPC-348
- Configuration Examples for 6VPE, page VPC-350

Configuring an MPLS VPN Using BGP: Example

The following example shows the configuration for an MPLS VPN using BGP on “vrf vpn1”:

```
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!
interface Loopback0
  ipv4 address 10.0.0.1 255.255.255.255
!
interface gigabitEthernet 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!
interface gigabitEthernet 0/1/0/1
  ipv4 address 10.0.0.1 255.0.0.0
!
router ospf 100
  area 100
    interface loopback0
    interface gigabitEthernet 0/1/0/1
  !
!
router bgp 100
  address-family vpnv4 unicast
  neighbor 10.0.0.3
    remote-as 100
    update-source Loopback0
    address-family vpnv4 unicast
  !
  vrf vpn1
    rd 100:1
    address-family ipv4 unicast
      redistribute connected
    !
    neighbor 10.0.0.1
      remote-as 200
      address-family ipv4 unicast
        as-override
        route-policy pass-all in
        route-policy pass-all out
      !
      advertisement-interval 5
    !
  !
!
mpls ldp
  route-id looback0
  interface gigabitEthernet 0/1/0/1
!
```

Configuring the Routing Information Protocol on the PE Router: Example

The following example shows the configuration for the RIP on the PE router:

```
vrf vpn1
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!

interface gigabitEthernet 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!

router rip
 vrf vpn1
  interface GigabitEthernet0/1/0/0
  !
  timers basic 30 90 90 120
  redistribute bgp 100
  default-metric 3
  route-policy pass-all in
 !
```

Configuring the PE Router Using EIGRP: Example

The following example shows the configuration for the Enhanced Interior Gateway Routing Protocol (EIGRP) on the PE router:

```
Router eigrp 10
 vrf VRF1
  address-family ipv4
   router-id 10.1.1.2
   default-metric 100000 2000 255 1 1500
   as 62
   redistribute bgp 2000
   interface Loopback0
   !
   interface GigabitEthernet0/6/0/0
```

Configuration Examples for MPLS VPN CSC

Configuration examples for the MPLS VPN CSC include:

- Configuring the Backbone Carrier Core: Examples, page VPC-349
- Configuring the Links Between CSC-PE and CSC-CE Routers: Examples, page VPC-349
- Configuring a Static Route to a Peer: Example, page VPC-350

Configuring the Backbone Carrier Core: Examples

Configuration examples for the backbone carrier core included in this section are as follows:

- Configuring VRFs for CSC-PE Routers: Example, page VPC-349
- Configuring the Links Between CSC-PE and CSC-CE Routers: Examples, page VPC-349

Configuring VRFs for CSC-PE Routers: Example

The following example shows how to configure a VPN routing and forwarding instance (VRF) for a CSC-PE router:

```
config
  vrf vpn1
   address-family ipv4 unicast
    import route-target 100:1
    export route-target 100:1
   end
```

Configuring the Links Between CSC-PE and CSC-CE Routers: Examples

This section contains the following examples:

- Configuring a CSC-PE: Example, page VPC-349
- Configuring a CSC-CE: Example, page VPC-349

Configuring a CSC-PE: Example

In this example, a CSC-PE router peers with a PE router, 10.1.0.2, in its own AS. It also has a labeled unicast peering with a CSC-CE router, 10.0.0.1.

```
config
    router bgp 2
        address-family vpnv4 unicast
        neighbor 10.1.0.2
            remote-as 2
            update-source loopback0
            address-family vpnv4 unicast
        vrf customer-carrier
            rd 1:100
            address-family ipv4 unicast
                allocate-label all
                redistribute static
        neighbor 10.0.0.1
            remote-as 1
            address-family ipv4 labeled-unicast
                route-policy pass-all in
                route-policy pass-all out
                as-override
end
```

Configuring a CSC-CE: Example

The following example shows how to configure a CSC-CE router. In this example, the CSC-CE router peers CSC-PE router 10.0.0.2 in AS 2.

```
config
    router bgp 1
        address-family ipv4 unicast
```

```
            redistribute ospf 200
            allocate-label all
        neighbor 10.0.0.2
            remote-as 2
            address-family ipv4 labeled-unicast
            route-policy pass-all in
            route-policy pass-all out
end
```

Configuring a Static Route to a Peer: Example

The following example shows how to configure a static route to an Inter-AS or CSC-CE peer:

```
config
 router static
  address-family ipv4 unicast
  10.0.0.2/32 40.1.1.1
end
```

Configuration Examples for 6VPE

Configuration examples for the MPLS VPN CSC include:

- Configuring an IPv6 Address Family Under VRF: Example, page VPC-350
- Configuring BGP for the Address Family VPNv6: Example, page VPC-350
- Configuring a PE-CE Protocol: Example, page VPC-351
- Configuring an Entire 6VPE Configuration: Example, page VPC-351

Configuring an IPv6 Address Family Under VRF: Example

The following example shows a standard configuration of an IPv6 address family under VRF:

```
configure
 vrf red
  address-family ipv6 unicast
   import route-target
    500:1
   !
   export route-target
    500:1
   !
  !
```

Configuring BGP for the Address Family VPNv6: Example

The following example shows the configuration for the address family VPNv6 under the PE peer:

```
configure
 router bgp 3
  address-family vpnv6 unicast
  !
   neighbor 192.168.254.3
   remote-as 3
   update-source Loopback0
   address-family ipv4 unicast
   !
   address-family vpnv44 unicast
   !
   address-family ipv6 labeled-unicast
   !
   address-family vpnv6 unicast
   !
  !
```

Configuring the Address Family IPv6 for the VRF Configuration Under BGP: Example

The following example shows the configuration for the address family IPv6 for the VRF configuration under BGP:

```
!
 vrf red
  address-family ipv6 unicast
  redistribute connected
  !
```

Configuring a PE-CE Protocol: Example

The following example shows the eBGP configuration of a PE-CE protocol:

```
  !
  neighbor 2001:db80:cafe:1::2
   remote-as 100
   address-family ipv6 unicast
   route-policy pass in
   route-policy pass out
```

Configuring an Entire 6VPE Configuration: Example

Two VPNs, which are named red and blue, are created across router2 and router4. The VRF red is for the user running IPv6 addressing in the network. The VRF blue is for the user running IPv4 addressing. 6VPE is implemented to carry the VPNv6 prefixes across to the other PE.

The following example shows the entire 6VPE configuration that includes the interface and VRF configurations of both PE routers across the route reflectors:

```
router2 (PE router)
interface GigabitEthernet0/0/1/3.1
 vrf red
 ipv4 address 192.3.1.1 255.255.255.0
 ipv6 address 2001:db80:cafe:1::1/64
 dot1q vlan 2
!

show run interface gigabitEthernet 0/0/1/3.2
interface GigabitEthernet0/0/1/3.2
 vrf blue
 ipv4 address 192.3.2.1 255.255.255.0
 dot1q vlan 3
!


vrf red
 address-family ipv4 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
 address-family ipv6 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
!
vrf blue
 address-family ipv4 unicast
  import route-target
   600:1
  !
  export route-target
   600:1
  !
 !


router bgp 3
 address-family ipv4 unicast
  network 3.3.3.3/32
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
  network 2001:db82:cafe:1::/64
  allocate-label all
 !
 address-family vpnv6 unicast
 !
 neighbor 192.168.253.4
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 vrf red
  rd 500:1
  address-family ipv4 unicast
```

```
    redistribute connected
   !
   address-family ipv6 unicast
    redistribute connected
   !
   neighbor 2001:db80:cafe:1::2
    remote-as 100
    address-family ipv6 unicast
     route-policy pass in
     route-policy pass out
    !
   !
  !
  vrf blue
   rd 600:1
   address-family ipv4 unicast
    redistribute connected
   !
  !
 !
```

```
router3 (RR)

router bgp 3
 bgp router-id 192.168.253.4
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
 !
 address-family vpnv6 unicast
 !
 neighbor-group all
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
   route-reflector-client
  !
  address-family vpnv4 unicast
   route-reflector-client
  !
  address-family ipv6 labeled-unicast
   route-reflector-client
  !
  address-family vpnv6 unicast
   route-reflector-client
  !
 !
 neighbor 192.168.253.1
  use neighbor-group all
 !
 neighbor 192.168.253.2
  use neighbor-group all
 !
 neighbor 192.168.253.3
  use neighbor-group all
 !
 neighbor 192.168.253.5
  use neighbor-group all
 !
 neighbor 192.168.253.6
  use neighbor-group all
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
 !
!

router4(PE router)

vrf red
 address-family ipv4 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
 address-family ipv6 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
!
vrf blue
 address-family ipv4 unicast
  import route-target
   600:1
  !
  export route-target
   600:1
  !
 !
!


router bgp 3
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
  network 2001:db84:beef:1::/64
  allocate-label all
 !
 address-family vpnv6 unicast
 !
 neighbor 192.168.253.4
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
 !
 vrf red
  rd 500:1
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
 !
 vrf blue
  rd 600:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
!
```

The following example displays the sample output for the entire 6VPE configuration:

```
show route vrf red ipv6

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

C    2001:db80:beef:1::/64 is directly connected,
      19:09:50, GigabitEthernet0/0/1/3.1
L    2001:db80:beef:1::1/128 is directly connected,
      19:09:50, GigabitEthernet0/0/1/3.1
B    2001:db80:cafe:1::/64
      [200/0] via ::ffff:192.168.253.3 (nexthop in vrf default), 07:03:40


show route vrf red ipv6

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

B    2001:db80:beef:1::/64
      [200/0] via ::ffff:192.168.253.6 (nexthop in vrf default), 07:04:14
C    2001:db80:cafe:1::/64 is directly connected,
         08:28:12, GigabitEthernet0/0/1/3.1
L    2001:db80:cafe:1::1/128 is directly connected,
         08:28:12, GigabitEthernet0/0/1/3.1
```

Additional References

For additional information, refer to the following documents:

Related Documents

Related Topic	Document Title
Cisco IOS XR L2VPN command reference document	*MPLS Virtual Private Network Commands on Cisco IOS XR Software module in Cisco IOS XR MPLS Configuration Guide*
Routing (BGP, EIGRP, OSPF, and RIP) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS XR Routing Command Reference*
Routing (BGP, EIGRP, OSPF, and RIP) configuration	*Cisco IOS XR Routing Configuration Guide*
MPLS LDP configuration: configuration concepts, task, and examples	*Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software*
MPLS Traffic Engineering Resource Reservation Protocol configuration: configuration concepts, task, and examples	*Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software* module in *Cisco IOS XR MPLS Command Reference*
Cisco CRS router getting started material	Cisco IOS XR *Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module in *Cisco IOS XR System Security Configuration Guide*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 1700	*Assigned Numbers*
RFC 1918	*Address Allocation for Private Internets*
RFC 1966	*BGP Route Reflectors: An Alternative to Full Mesh iBGP*
RFC 2283	*Multiprotocol Extensions for BGP-4*
RFC 2547	*BGP/MPLS VPNs*
RFC 2842	*Capabilities Advertisement with BGP-4*
RFC 2858	*Multiprotocol Extensions for BGP-4*
RFC 3107	*Carrying Label Information in BGP-4*

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

INDEX

HC	Cisco IOS XR Interface and Hardware Component Configuration Guide
IC	Cisco IOS XR IP Addresses and Services Configuration Guide
MCC	Cisco IOS XR Multicast Configuration Guide
MNC	Cisco IOS XR System Monitoring Configuration Guide
MPC	Cisco IOS XR MPLS Configuration Guide
QC	Cisco IOS XR Modular Quality of Service Configuration Guide
RC	Cisco IOS XR Routing Configuration Guide
SC	Cisco IOS XR System Security Configuration Guide
SMC	Cisco IOS XR System Management Configuration Guide
VPC	Cisco IOS XR Virtual Private Network Configuration Guide