Preface

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

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

Changes to This Document

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>OL-24669-01</td>
<td>April 2011</td>
<td>Initial release of this document.</td>
</tr>
</tbody>
</table>

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:


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.
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.

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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.4.0</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 3.4.1</td>
<td>Support was added for:</td>
</tr>
<tr>
<td></td>
<td>• Virtual Circuit Connection Verification (VCCV) on L2VPN</td>
</tr>
<tr>
<td></td>
<td>• Layer 2 VPN (L2VPN) Quality of Service (QoS) for Ethernet-over-MPLS (EoMPLS)</td>
</tr>
<tr>
<td>Release 3.5.0</td>
<td>Support was added for:</td>
</tr>
<tr>
<td></td>
<td>• EoMPLS Inter-AS mode</td>
</tr>
<tr>
<td></td>
<td>• Mac-in-Mac protocol</td>
</tr>
<tr>
<td>Release 3.6.0</td>
<td>Support was added for:</td>
</tr>
<tr>
<td></td>
<td>• Ethernet Remote Port Shutdown</td>
</tr>
<tr>
<td></td>
<td>• Preferred Tunnel Path</td>
</tr>
<tr>
<td>Release 3.7.0</td>
<td>Support was added for ATM over MPLS (ATMoMPLS) with Layer 2VPN capability.</td>
</tr>
<tr>
<td>Release 3.8.0</td>
<td>Support was added for QinQ mode and QinAny mode for EoMPLS.</td>
</tr>
<tr>
<td>Release 3.9.0</td>
<td>Support was added for the Generic Routing Encapsulation (GRE) feature.</td>
</tr>
</tbody>
</table>
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:

- L2VPN Overview, page VPC-16
- ATMoMPLS with L2VPN Capability, page VPC-17
- Virtual Circuit Connection Verification on L2VPN, page VPC-18
- Ethernet over MPLS, page VPC-18
- Quality of Service, page VPC-22
- High Availability, page VPC-23
- Preferred Tunnel Path, page VPC-23
- Any Transport over MPLS, page VPC-25

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 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-17
- Layer 2 Local Switching Overview, page VPC-17
- ATM Adaptation Layer 5, page VPC-17

**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 between interfaces of different types 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-18
- Ethernet Remote Port Shutdown, page VPC-19
- VLAN Mode, page VPC-19
- Inter-AS Mode, page VPC-20
- QinQ Mode, page VPC-21
- QinAny Mode, 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.
Ethernet Remote Port Shutdown

Ethernet remote port shutdown provides a mechanism for the detection and propagation of remote link failure for port mode EoMPLS on a Cisco CRS-1 line card. This lets a service provider edge router on the local end of an Ethernet-over-MPLS (EoMPLS) pseudowire detect a cross-connect or remote link failure and cause the shutdown of the Ethernet port on the local customer edge router. Shutting down the Ethernet port on the local customer edge router prevents or mitigates a condition where that router would otherwise lose data by forwarding traffic continuously to the remote failed link, especially if the link were configured as a static IP route (see Figure 2).

To enable this functionality, see the l2transport propagate command in Cisco IOS XR MPLS Command Reference.

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 3, 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.
Implementing MPLS Layer 2 VPNs

Information About Implementing L2VPN

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 4** illustrates MPLS over Inter-AS with a basic double AS topology with iBGP/LDP in each AS.
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 5 illustrates QinQ using VC type 4.
QinAny Mode

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

Quality of Service

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

L2VPN technology requires that QoS functionality on PE routers be strictly L2-payload-based on the edge-facing interfaces (also known as attachment circuits). Figure 6 illustrates L2 and L3 QoS service policies in a typical L2VPN network.

Figure 7 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.
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.
Generic Routing Encapsulation Support for L2VPN

Generic Routing Encapsulation (GRE) is a tunneling protocol that can encapsulate many types of packets to enable data transmission using a tunnel.

Ethernet over MPLS Forwarding Using GRE Tunnels

This section describes the working of the Ethernet over MPLS (EoMPLS) over GRE tunnels. The following description assumes that the GRE tunnels connect two PE routers, and Layer 2 circuits exist between the PE and CE routers.

Ingress of Encapsulation Router

To enable L2VPN forwarding over GRE tunnels, the targeted Label Distribution Protocol (LDP) neighbor session (established over the GRE tunnel interface) enables exchanging virtual circuit labels between the PE routers. LDP also installs an implicit null label to be used across the GRE tunnels. The implicit null label is a label with special semantics that an LDP can bind to an address prefix. The Layer 2 forwarding tracks the GRE tunnel or addresses, depending on the GRE tunnel.

Egress of Encapsulation Router

On the egress side of the encapsulation PE router, the following events occur:

- The Layer 2 packet passes through the regular L2VPN processing based on the local label imposed by the ingress side.
- The Layer 2 packet is encapsulated with the GRE header and outer IP header.
- The packet is transmitted based on the outer IP header information.

Ingress of Decapsulation Router

When the decapsulation router (remote PE) receives the GRE encapsulated Layer 2 packet, the following events occur:

- The packet is processed based on the outer IP header information.
- The packet is decapsulated to retrieve the inner LDP packet.

Further processing of the packet is based on the Layer 2 payload. The packet is handed over to the egress line card (that hosts the Layer 2 circuit) based on the VC label forwarding information programmed by the Layer 2 FIB.

Egress of Decapsulation Router

On the egress side of the decapsulation router, the following events occur:

- The forwarding occurs based on the Layer 2 VC label on the packet.
- A VC label lookup identifies the correct Layer 2 circuit to be used, to forward the packet towards the CE router that hosts the Layer 2 destination.
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.

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, 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:

- Control Word Processing, page VPC-25

Control Word Processing

The control word contains forward explicit congestion notification (FECN), backward explicit congestion notification (BECN).

Control word is mandatory for the following:

- ATM AAL5
How to Implement L2VPN

This section describes the tasks required to implement L2VPN:

- Configuring an Interface or Connection for L2VPN, page VPC-26
- Configuring Static Point-to-Point Cross-Connects, page VPC-28
- Configuring Dynamic Point-to-Point Cross-Connects, page VPC-30
- Configuring Inter-AS, page VPC-32
- Configuring L2VPN Quality of Service, page VPC-32
- Configuring Preferred Tunnel Path, page VPC-36

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>interface type interface-path-id</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/0/0/0</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>l2transport</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-if)# l2transport</td>
</tr>
</tbody>
</table>
### Step 4

**Command or Action**

exit

**Purpose**

Exits the current configuration mode.

**Example:**

```
RP/0/RP0/CPU0:router(config-if-l2)# exit
```

### Step 5

**Command or Action**

interface type interface-path-id

**Purpose**

Enters interface configuration mode and configures an interface.

**Example:**

```
RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0
```

### Step 6

**Command or Action**

dot1q native vlan vlan ID

**Purpose**

Assigns the native VLAN ID of a physical interface trunking 802.1Q VLAN traffic.

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# dot1q vlan 1
```

### Step 7

**Command or Action**

end

or

commit

**Purpose**

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.

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# end
```

or

```
RP/0/RP0/CPU0:router(config-if)# commit
```
Configuring Static Point-to-Point Cross-Connects

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

Please consider the following 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. interface type interface-path-id
6. neighbor ip-address pw-id pseudowire-id
7. mpls static label local {value} remote {value}
8. end
   or commit
9. show l2vpn xconnect group group name

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> 12vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# 12vpn</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 3</strong> xconnect group group name</td>
<td>Enters the name of the cross-connect group.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group vlan_grp_1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> p2p xconnect name</td>
<td>Enters a name for the point-to-point cross-connect.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-vc)# p2p vlan1</td>
<td></td>
</tr>
</tbody>
</table>
| **Step 5** interface type interface-path-id | Specifies the interface type ID. The choices are:  
- GigabitEthernet: GigabitEthernet/IEEE 802.3 interfaces.  
- TenGigE: TenGigabitEthernet/IEEE 802.3 interfaces. |
| Example: RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0.1 | |
| **Step 6** neighbor ip-address pw-id pseudowire-id | Configures the pseudowire segment for the cross-connect. Optionally, you can disable the control word or set the transport-type to "Ethernet" or "VLAN". |
| Example: RP/0/RP0/CPU0:router(config-l2vpn-xc-xc-p2p)# neighbor 2.2.2.2 pw-id 2000 | |
| **Step 7** mpls static label local {value} remote {value} | Configures local and remote label ID values. |
| Example: RP/0/RP0/CPU0:router(config-l2vpn-xc-xc-p2p-pw)# mpls static label local 699 remote 890 | |
Implementing MPLS Layer 2 VPNs

How to Implement L2VPN

Pseudowire-Switched Private VLAN Configuration

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

Note

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 8**
| end
| or
| 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 9**
| `show l2vpn xconnect group group-name`
| Displays the name of the Point-to-Point cross-connect group you created.
| Example:
| RP/0/RP0/CPU0:show l2vpn xconnect group p2p

**Configuring Dynamic Point-to-Point Cross-Connects**

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **SUMMARY STEPS**
| 1. configure
| 2. l2vpn
| 3. xconnect group group-name
| 4. p2p xconnect-name
| 5. interworking ipv4
| 6. interface type interface-path-id
| 7. neighbor ip-address pw-id pseudowire-id
| 8. end
| or
| commit

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.

Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.

Example:
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# e
or
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit

Step 9

Example:
RP/0/RP0/CPU0:show l2vpn xconnect group p2p
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters the configuration mode.</td>
</tr>
<tr>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><code>l2vpn</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters the name of the cross-connect group.</td>
</tr>
<tr>
<td><code>xconnect group group-name</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group grp_1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters a name for the point-to-point cross-connect.</td>
</tr>
<tr>
<td><code>p2p xconnect-name</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p vlan1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Configure the interworking for IPv4 network.</td>
</tr>
<tr>
<td><code>interworking ipv4</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-xc)# interworking ipv4</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Specifies the interface type ID. The choices are:</td>
</tr>
<tr>
<td><code>interface type interface-path-id</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# interface GigabitEthernet0/0/0/0.1</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• GigabitEthernet: GigabitEthernet/IEEE 802.3 interfaces.</td>
</tr>
<tr>
<td></td>
<td>• TenGigE: TenGigabitEthernet/IEEE 802.3 interfaces.</td>
</tr>
</tbody>
</table>
Implementing MPLS Layer 2 VPNs

How to Implement L2VPN

VPC-32
Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

OL-24669-01

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-28 and “Configuring Dynamic Point-to-Point Cross-Connects” section on page MPC-30) 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 and ATM sub-interfaces.

Restrictions

The l2transport command cannot be used with any IP address, L3, or CDP configuration.

### Command or Action

<table>
<thead>
<tr>
<th>Step 7</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>neighbor ip-address pw-id pseudowire-id</td>
<td>Configures the pseudowire segment for the cross-connect. Optionally, you can disable the control word or set the transport-type to &quot;Ethernet&quot; or &quot;vlan&quot;.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 2.2.2.2 pw-id 2000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 8</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>end or commit</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# end</td>
<td>When you issue the end command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td></td>
<td>or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# commit</td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters the configuration mode.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th><strong>Step 2</strong> interface type interface-path-id.subinterface l2transport</th>
<th>Configures an interface or connection for L2 switching and specifies the interface attachment circuit.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0/1</td>
<td></td>
</tr>
</tbody>
</table>

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router
Configuring an L2VPN Quality of Service Policy in VLAN Mode

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

Command or Action | Purpose
---|---
**Step 3**

```
service-policy [input | output] [policy-map-name]
```

Attaches a QoS policy to an input or output interface to be used as the service policy for that interface.

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# service-policy input servpol1
```

**Step 4**

```
end
```
or

```
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.

**SUMMARY STEPS**

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

**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 GigabitEthernet0/0/0.1 l2transport”).
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters the configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> interface</td>
<td>Configures an interface or connection for L2 switching.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0/0.1 l2transport</td>
<td>Note In VLAN Mode, you must enter the l2transport keyword on the same line as the interface.</td>
</tr>
<tr>
<td><strong>Step 3</strong> service-policy</td>
<td>Attaches a QoS policy to an input or output interface to be used as the service policy for that interface.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# service-policy input servpol1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> end or commit</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# end or RP/0/RP0/CPU0:router(config-if)# commit</td>
<td>• When you issue the end command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>– Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>– Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
</tbody>
</table>
Configuring Preferred Tunnel Path

This procedure describes how to configure a preferred tunnel path.

**Note**
The tunnel used for the preferred path configuration is an MPLS Traffic Engineering (MPLS-TE) tunnel.

**SUMMARY STEPS**

1. configure
2. l2vpn
3. pw-class *name*
4. encapsulation mpls
5. preferred-path [interface] [tunnel-te value] [fallback disable]
6. end
   or
   commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters the configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> pw-class <em>name</em></td>
<td>Configures the pseudowire class name.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-12vpn)# pw-class path1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> encapsulation mpls</td>
<td>Configures the pseudowire encapsulation to MPLS.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-12vpn-pwc)# encapsulation mpls</td>
<td></td>
</tr>
</tbody>
</table>
In the following 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 the following configuration examples:

- L2VPN Interface Configuration: Example, page VPC-38
- Point-to-Point Cross-connect Configuration: Examples, page VPC-38
- Inter-AS: Example, page VPC-38
- L2VPN Quality of Service: Example, page VPC-40
- Preferred Path: Example, page VPC-40
- AToM Cross Connect Configuration: Example, page VPC-40
- AToM Cross Connect Configuration: Example, page VPC-40
- Configuring L2VPN over GRE Tunnels: Example, page VPC-41
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 GigabitEthernet0/0/0/1
  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 GigabitEthernet0/0/0/1
  neighbor 2.2.1.1 pw-id 1commit
```

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!
```
Implementing MPLS Layer 2 VPNs

Configuration Examples for L2VPN

VPC-39

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

OL-24669-01

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 GigabitEthernet0/0/0.1
      neighbor 20.0.0.5 pw-id 101
      !
    p2p ac2ac2
      interface GigabitEthernet0/0/0.2
      neighbor 20.0.0.5 pw-id 102
      !
    p2p ac2ac3
      interface GigabitEthernet0/0/0.3
      neighbor 20.0.0.5 pw-id 103
      !
    p2p ac2ac4
      interface GigabitEthernet0/0/0.4
      neighbor 20.0.0.5 pw-id 104
      !
    p2p ac2ac5
      interface GigabitEthernet0/0/0.5
      neighbor 20.0.0.5 pw-id 105
      !
    p2p ac2ac6
      interface GigabitEthernet0/0/0.6
      neighbor 20.0.0.5 pw-id 106
      !
    p2p ac2ac7
      interface GigabitEthernet0/0/0.7
      neighbor 20.0.0.5 pw-id 107
      !
    p2p ac2ac8
      interface GigabitEthernet0/0/0.8
      neighbor 20.0.0.5 pw-id 108
      !
    p2p ac2ac9
      interface GigabitEthernet0/0/0.9
      neighbor 20.0.0.5 pw-id 109
      !
    p2p ac2ac10
      interface GigabitEthernet0/0/0.10
Implementing MPLS Layer 2 VPNs

Configuration Examples for L2VPN

neighbor 20.0.0.5 pw-id 110
!
!
mls 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 value fallback disable

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
        !
    !
Implementing MPLS Layer 2 VPNs

Configuration Examples for L2VPN

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 800pw-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
Additional References

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

Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XR L2VPN command reference document</td>
<td><strong>MPLS Virtual Private Network Commands on Cisco IOS XR Software module</strong></td>
</tr>
<tr>
<td>MPLS VPN-related commands</td>
<td>in <strong>Cisco IOS XR MPLS Command Reference</strong></td>
</tr>
<tr>
<td>MPLS Layer 2 VPNs</td>
<td><strong>Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software module</strong></td>
</tr>
<tr>
<td>MPLS Layer 3 VPNs</td>
<td>in <strong>Cisco IOS XR MPLS Configuration Guide</strong></td>
</tr>
<tr>
<td>MPLS VPNs over IP Tunnels</td>
<td><strong>MPLS VPNs over IP Tunnels on Cisco IOS XR Software module</strong></td>
</tr>
<tr>
<td>Cisco CRS router getting started material</td>
<td><strong>Cisco IOS XR Getting Started Guide</strong></td>
</tr>
<tr>
<td>Information about user groups and task IDs</td>
<td><strong>Configuring AAA Services on Cisco IOS XR Software module</strong> of <strong>Cisco IOS XR System Security Configuration Guide</strong></td>
</tr>
</tbody>
</table>

Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Not all supported standards are listed.</td>
<td>—</td>
</tr>
</tbody>
</table>

MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <strong><a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></strong></td>
</tr>
</tbody>
</table>
RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 3931</td>
<td>Layer Two Tunneling Protocol - Version 3 (L2TPv3)</td>
</tr>
<tr>
<td>RFC 4447</td>
<td>Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP), April 2006</td>
</tr>
<tr>
<td>RFC 4448</td>
<td>Encapsulation Methods for Transport of Ethernet over MPLS Networks, April 2006</td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.8.0</td>
<td>This feature was introduced. Support for the bridging functionality feature (VPLS based) and pseudowire redundancy was added.</td>
</tr>
<tr>
<td>Release 3.9.0</td>
<td>The following features were added:</td>
</tr>
<tr>
<td></td>
<td>• Blocking unknown unicast flooding.</td>
</tr>
<tr>
<td></td>
<td>• Disabling MAC flush.</td>
</tr>
<tr>
<td>Release 4.0</td>
<td>The following features were added:</td>
</tr>
<tr>
<td></td>
<td>• H-VPLS with MPLS Access pseudowire</td>
</tr>
<tr>
<td></td>
<td>• H-VPLS with Ethernet Access</td>
</tr>
<tr>
<td></td>
<td>• MAC Address withdrawal</td>
</tr>
<tr>
<td>Release 4.0.1</td>
<td>Support for the BGP Autodiscovery with LDP Signaling feature was added.</td>
</tr>
<tr>
<td>Release 4.1.0</td>
<td>Support for Pseudowire Headend feature was added.</td>
</tr>
</tbody>
</table>
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 Cisco CRS-1 router supports multiple types of attachment circuits in a bridge domain.
- The line card requires ternary content addressable memory (TCAM) Carving configuration. The Cisco CRS-1 router however, does not require the 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.
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. The Cisco CRS-1 router supports only the Ethernet port mode and the 802.1q VLAN 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, page VPC-47
- VPLS for an MPLS-based Provider Core, page VPC-48
- Hierarchical VPLS, page VPC-48
- VPLS Discovery and Signaling, page VPC-50
- Bridge Domain, page VPC-53
- MAC Address-related Parameters, page VPC-53
- LSP Ping over VPWS and VPLS, page VPC-56
- Pseudowire Redundancy for P2P AToM Cross-Connects, page VPC-57
- Pseudowire Headend, page VPC-57

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 8 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 8  Ethernet Access for H-VPLS

H-VPLS with PW-access

Figure 9 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 9.

Figure 9  PW access for H-VPLS
VPLS Discovery and Signaling

VPLS is a Layer 2 multipoint service and it emulates a LAN service across a WAN. VPLS enables service providers to interconnect several LAN segments over a packet-switched network and make them behave as a single LAN. Service providers can provide a native Ethernet access connection to customers using VPLS.

The VPLS control plane consists of two important components, autodiscovery and signaling:

- VPLS Autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS Autodiscovery enables each VPLS PE router to discover other provider edge (PE) routers that are part of the same VPLS domain.
- Once the PEs are discovered, pseudowires (PWs) are signaled and established across pairs of PE routers, forming a full mesh of PWs across PE routers in a VPLS domain.

**Figure 10  VPLS Autodiscovery and Signaling**

<table>
<thead>
<tr>
<th>L2-VPN</th>
<th>Multipoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery</td>
<td>BGP</td>
</tr>
<tr>
<td>Signaling Protocol</td>
<td>LDP</td>
</tr>
<tr>
<td>Tunneling Protocol</td>
<td>MPLS</td>
</tr>
</tbody>
</table>

**BGP-based VPLS Autodiscovery**

An important aspect of VPN technologies, including VPLS, is the ability of network devices to automatically signal information to other devices, about any association with a particular VPN. Autodiscovery requires this information to be distributed to all members of a VPN. VPLS is a multipoint mechanism for which BGP is well-suited.

BGP-based VPLS autodiscovery eliminates the need to manually provision VPLS neighbors. VPLS autodiscovery enables each VPLS PE router to discover other provider edge (PE) routers that are part of the same VPLS domain. VPLS Autodiscovery also tracks occurrences when PE routers are added to, or removed from, the VPLS domain. When the discovery process is complete, each PE router has the information required to setup VPLS pseudowires (PWs).

**BGP Auto Discovery With BGP Signaling**

The implementation of VPLS in a network requires the establishment of a full mesh of PWs between the provider edge (PE) routers. The PWs can be signaled using BGP signaling.
The BGP signaling and autodiscovery scheme has these components:

- A means by which a PE can learn which remote PEs are members of a given VPLS. This process is known as autodiscovery.
- A means by which a PE can learn about the pseudowire label that is expected by a given remote PE for a given VPLS. This process is known as signaling.

The BGP Network Layer Reachability Information (NLRI) takes care of both these components simultaneously. The NLRI generated by a given PE contains necessary information required by other PEs. These components enable the automatic setup of a full mesh of pseudowires for each VPLS, without having to manually configure those pseudowires on each PE.

**NLRI Format for VPLS with BGP AD and Signaling**

*Figure 12* shows the NLRI format for VPLS with BGP AD and Signaling.

**BGP Auto Discovery With LDP Signaling**

Signaling of pseudowires requires exchange of information between two endpoints. Label Distribution Protocol (LDP) is better suited for point-to-point signaling. The signaling of pseudowires, between provider edge devices, uses targeted LDP sessions to exchange label values and attributes, and configure the pseudowires.
A PE router advertises an identifier through BGP for each VPLS instance. This identifier is unique within the VPLS instance and acts like a VPLS ID. The identifier enables the PE router, receiving the BGP advertisement, to identify the VPLS associated with the advertisement, and import it to the correct VPLS instance. In this manner, for each VPLS, a PE router learns which other PE routers are members of the VPLS.

The LDP protocol is used to configure a pseudowire to all other PE routers. The FEC 129 standard is used for signaling. The information carried by FEC 129 includes the VPLS ID, the Target Attachment Individual Identifier (TAII) and the Source Attachment Individual Identifier (SAII).

The LDP advertisement also contains the inner label or VPLS label that is expected for incoming traffic over the pseudowire. This enables the LDP peer to identify the VPLS instance with which the pseudowire is to be associated, and the label value that it is expected to use when sending traffic on that pseudowire.

**NLRI and Extended Communities**

Figure 12 depicts NLRI and extended communities.
Interoperability Between Cisco IOS XR and Cisco IOS on VPLS LDP Signaling

The Cisco IOS Software encodes the NLRI length in the first 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 2 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-54
- MAC Address-based Forwarding, page VPC-54
- MAC Address Source-based Learning, page VPC-54
- MAC Address Aging, page VPC-54
- MAC Address Limit, page VPC-55
- MAC Address Withdrawal, page VPC-55
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 2.

### Table 2  MAC Address Limit Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit flood</td>
<td>Discards the new MAC addresses.</td>
</tr>
<tr>
<td>Limit no-flood</td>
<td>Discards the new MAC addresses. Flooding of unknown unicast packets is disabled.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

**Note**

On the Cisco CRS-1 router, MAC address limit action is supported only on the ACs and not on core pseudowires.

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 CRS Router*.

**VPLS Scalability and Performance Targets**

The Cisco CRS-1 router employs the ternary content addressable memory (TCAM) to meet the performance and scalable targets over VPLS.

Table 3 describes the scalability and performance targets for the Cisco CRS-1 router.

<table>
<thead>
<tr>
<th>Performance</th>
<th>Scalability Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum bridge domains per Line Card</td>
<td>1024</td>
</tr>
<tr>
<td>Maximum bridge domains per system</td>
<td>1024</td>
</tr>
<tr>
<td>Maximum MACs per bridge domain</td>
<td>15999</td>
</tr>
<tr>
<td>Maximum MACs per Line Card</td>
<td>65536</td>
</tr>
<tr>
<td>Maximum MACs per system</td>
<td>65536</td>
</tr>
<tr>
<td>Maximum attachment circuits per bridge domain</td>
<td>4085</td>
</tr>
<tr>
<td>Maximum pseudowires per bridge domain</td>
<td>256</td>
</tr>
<tr>
<td>Maximum pseudowires per system</td>
<td>16340</td>
</tr>
</tbody>
</table>
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.

It is recommended to enable pseudowire status time length value (TLV) for optimal switchover performance.

**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.

---

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 (Cisco CRS 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 15** PWHE example

![PWHE Diagram](image)

Note that 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 16 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 16  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.

How to Implement Virtual Private LAN Services

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

- Configuring a Bridge Domain, page VPC-58
- Configuring a Layer 2 Virtual Forwarding Instance, page VPC-77
- Configuring the MAC Address-related Parameters, page VPC-89
- Configuring VPLS with BGP Autodiscovery and Signaling, page VPC-101
- Configuring VPLS with BGP Autodiscovery and LDP Signaling, page VPC-104
- Configuring Pseudowire Headend, page VPC-107

Configuring a Bridge Domain

These topics describe how to configure a bridge domain:

- Creating a Bridge Domain, page VPC-59
- Configuring a Pseudowire, page VPC-60
- Configuring an Access Pseudowire, page VPC-63
- Associating Members with a Bridge Domain, page VPC-72
- Configuring Bridge Domain Parameters, page VPC-74
- Disabling a Bridge Domain, page VPC-75
- Configuring a Layer 2 Virtual Forwarding Instance, page VPC-77

**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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>l2vpn</code></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)#</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>bridge group bridge-group-name</code></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco</code></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg)#</code></td>
<td></td>
</tr>
</tbody>
</table>
## 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`

### Command or Action | Purpose
--- | ---
**Step 4**

`bridge-domain bridge-domain-name`

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#
```

**Step 5**

`end`

or

`commit`

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# end
or
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# commit
```

Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.

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.
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>12vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
</tbody>
</table>
| Example: | RP/0/RP0/CPU0:router(config)# 12vpn  
RP/0/RP0/CPU0:router(config-12vpn)# | |
| Step 3 | bridge group bridge group name | Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain. |
| Example: | RP/0/RP0/CPU0:router(config-12vpn)# bridge group  
cisco  
RP/0/RP0/CPU0:router(config-12vpn-bg)# | |
| Step 4 | bridge-domain bridge-domain name | Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode. |
| Example: | RP/0/RP0/CPU0:router(config-12vpn-bg)# bridge-domain  
abc  
RP/0/RP0/CPU0:router(config-12vpn-bg-bd)# | |
| Step 5 | vfi {vfi-name} | Configures the virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode. |
| Example: | RP/0/RP0/CPU0:router(config-12vpn-bg-bd)# vfi v1  
RP/0/RP0/CPU0:router(config-12vpn-bg-bd-vfi)# |  
- Use the vfi-name argument to configure the name of the specified virtual forwarding interface. |
| Step 6 | exit | Exits the current configuration mode. |
| Example: | RP/0/RP0/CPU0:router(config-12vpn-bg-bd-vfi)# exit  
RP/0/RP0/CPU0:router(config-12vpn-bg-bd)# | |
### Command or Action

**Step 7**

```
neighbor (A.B.C.D) {pw-id value}
```

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-pw)#
```

### Purpose

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/RP0/CPU0:router(config-l2vpn-bg-bd-pw)# end
```

or

```
RP/0/RP0/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)?
  
  - 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 Access Pseudowire

Perform this task to configure an access pseudowire under a bridge domain.

SUMMARY STEPS

1. configure
2. l2vpn
3. bridge group `bridge group name`
4. bridge-domain `bridge-domain name`
5. interface `type interface-path-id`
6. neighbor `{A.B.C.D} `{pw-id value}`
7. end
   or
   commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2vpn RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group <code>bridge group name</code></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> bridge-domain <code>bridge-domain name</code></td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> interface <code>type interface-path-id</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0 RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#</td>
<td></td>
</tr>
</tbody>
</table>
## How to Implement Virtual Private LAN Services

### Command or Action

<table>
<thead>
<tr>
<th>Step 6</th>
<th>exit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# exit RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 7</th>
<th>neighbor {A.B.C.D} {pw-id value}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# neighbor 10.1.1.2 pw-id 1000 RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-pw)#</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 8</th>
<th>end or commit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-pw)# end or RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-pw)# commit</td>
</tr>
</tbody>
</table>

### Purpose

- **Step 6**: Exits the current configuration mode.
- **Step 7**: 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. The range is 1 to 4294967295.
- **Step 8**: 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 Status TLV

When a pseudowire is setup, label distribution protocol (LDP) determines the method for signaling pseudowire status. Cisco IOS-XR provides a configuration option that allows you to enable pseudowire status type length value (TLV).

Note

Unless pseudowire status TLV is explicitly enabled under L2VPN configuration, the default signaling method is Label Withdrawal. Pseudowire status TLV must be enabled on both local and remote PEs. If only one provider edge router is configured with the `pw-status tlv` command, then label withdrawal method is used.

Perform this task to enable pseudowire status TLV.

**SUMMARY STEPS**

1. `configure`
2. `l2vpn`
3. `pw-status tlv`
4. `end`
   or
5. `commit`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>l2vpn</code></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)#</code></td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step 3</th>
<th>pw-status tlv</th>
<th>Enables pseudowire status TLV.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# pw-status tlv</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 4</th>
<th>end or commit</th>
<th>Saves configuration changes.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# end or commit</td>
<td></td>
</tr>
</tbody>
</table>

- 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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config)# l2vpn RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> xconnect group group name</td>
<td>Enters the name of the cross-connect group.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group A RP/0/RP0/CPU0:router(config-l2vpn-xc)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> p2p xconnect name</td>
<td>Enters a name for the point-to-point cross-connect.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p rtrX_to_rtrY RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> neighbor ip-address pw-id number</td>
<td>Configures the pseudowire segment for the cross-connect.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 1.1.1.1 pw-id 2 RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)#</td>
<td></td>
</tr>
</tbody>
</table>
### Implementing Virtual Private LAN Services

**How to Implement Virtual Private LAN Services**

**VPC-68**

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

### Command or Action

<table>
<thead>
<tr>
<th>Step 6</th>
<th>backup neighbor ip-address pw-id number</th>
</tr>
</thead>
</table>

**Example:**

RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# backup neighbor 1.1.1.1 pw-id 2

### Purpose

Configures the backup pseudowire for the point-to-point neighbor.

### Command or Action

<table>
<thead>
<tr>
<th>Step 7</th>
<th>end</th>
</tr>
</thead>
</table>

**Example:**

RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# end

or

RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# commit

### Purpose

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config)# l2vpn RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> pw-class class_1</td>
<td>Configures the pseudowire class name.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn)# pw-class class_1 RP/0/RP0/CPU0:router(config-l2vpn-pwc)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> backup disable delay seconds</td>
<td>Specifies how long a backup pseudowire virtual circuit (VC) should wait before resuming operation after the primary pseudowire VC becomes nonfunctional.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-pwc)# backup disable delay 20 RP/0/RP0/CPU0:router(config-l2vpn-pwc)#</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>exit</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-pwc)# exit</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>xconnect group group name</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group A</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>p2p xconnect name</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)#</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>neighbor ip-address pw-id number</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)#</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>pw-class class_1</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class class_1</td>
</tr>
</tbody>
</table>
### Command or Action

**Step 10**

```
backup neighbor ip-address pw-id number
```

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# backup neighbor 1.1.1.1 pw-id 2
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)#
```

**Purpose**

Configures the backup pseudowire for the point-to-point neighbor.

**Step 11**

```
end
```

or

```
commit
```

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# end
```

or

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw-backup)# commit
```

**Purpose**

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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>l2vpn</code></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td><code>bridge group bridge-group-name</code></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>bridge-domain bridge-domain-name</code></td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
</tbody>
</table>
## Command or Action

### Step 5

**interface** type interface-path-id

**Example:**

```plaintext
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/4/0/0
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)#
```

Enters interface configuration mode and adds an interface to a bridge domain that allows packets to be forwarded and received from other interfaces that are part of the same bridge domain.

### Step 6

**static-mac-address** {MAC-address}

**Example:**

```plaintext
RP/0/RP0/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:**

```plaintext
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# end  
RP/0/RP0/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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td><code>l2vpn</code></td>
<td>Enters l2vpn configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
<td>Enters l2vpn configuration mode.</td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)#</code></td>
<td>Enters l2vpn configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td></td>
</tr>
<tr>
<td><code>bridge group bridge group name</code></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco</code></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg)#</code></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>bridge-domain bridge-domain name</code></td>
<td>Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc</code></td>
<td>Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</code></td>
<td>Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.</td>
</tr>
</tbody>
</table>
### Disabling a Bridge Domain

Perform this task to disable a bridge domain. When a bridge domain is disabled, all VFI s that are associated with the bridge domain are disabled. You are still able to attach or detach members to the bridge domain and the VFI s 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong> flooding disable</td>
<td>Configures flooding for traffic at the bridge domain level or at the bridge port level.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# flooding disable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> mtu bytes</td>
<td>Adjusts the maximum packet size or maximum transmission unit (MTU) size for the bridge domain.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mtu 1000</td>
<td>• Use the bytes argument to specify the MTU size, in bytes. The range is from 64 to 65535.</td>
</tr>
<tr>
<td><strong>Step 7</strong> end or commit</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# end or commit</td>
<td>• When you issue the end command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td></td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
</tr>
<tr>
<td></td>
<td>– Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>– Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>– Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
</tbody>
</table>
## How to Implement Virtual Private LAN Services

6. **end**
   or
   **commit**

### Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2vpn RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group bridge-group-name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> bridge-domain bridge-domain-name</td>
<td>Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
</tbody>
</table>
These topics describe how to configure a Layer 2 virtual forwarding instance (VFI):

- **Adding the Virtual Forwarding Instance Under the Bridge Domain**, page VPC-78
- **Associating Pseudowires with the Virtual Forwarding Instance**, page VPC-79
- **Associating a Virtual Forwarding Instance to a Bridge Domain**, page VPC-81
- **Attaching Pseudowire Classes to Pseudowires**, page VPC-83
- **Configuring Any Transport over Multiprotocol Pseudowires By Using Static Labels**, page VPC-85
- **Disabling a Virtual Forwarding Instance**, page VPC-87
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group bridge group name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> bridge-domain bridge-domain name</td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
</tbody>
</table>
## Implementing Virtual Private LAN Services

### How to Implement Virtual Private LAN Services

**VPC-79**

**Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router**

---

### 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`

---

**Step 5**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vfi {vfi name}</code></td>
<td>Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
```

**Step 6**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>end</code> or <code>commit</code></td>
<td>Saves configuration changes.</td>
</tr>
</tbody>
</table>

- 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.

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-vpn)# end
or
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-vpn)# commit
```
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2vpn RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group bridge group name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> bridge-domain bridge-domain name</td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> vfi (vfi name)</td>
<td>Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi vl RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#</td>
<td></td>
</tr>
</tbody>
</table>
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}

**Step 6**

**Command or Action**

**neighbor A.B.C.D (pw-id value)**

**Example:**

RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#
neighbor 10.1.1.2 pw-id 1000
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#

**Step 7**

**end**
or

**commit**

**Example:**

RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end
or
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit

**Purpose**

Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).

- Use the *A.B.C.D* argument to specify the IP address of the cross-connect peer.
- Use the *pw-id* keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.

**Step 6**

**Summary**

- 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.
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group bridge group name</td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bd)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> bridge-domain bridge-domain name</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bd)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> vfi vfi name</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bd-vfi)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> neighbor A.B.C.D (pw-id value)</td>
<td>Add an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bd-vfi)#</td>
<td>Use the A.B.C.D argument to specify the IP address of the cross-connect peer.</td>
</tr>
<tr>
<td>neighbor 10.1.1.2 pw-id 1000</td>
<td>Use the pw-id keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bd-vfi-pw)#</td>
<td></td>
</tr>
</tbody>
</table>
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

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
static-mac-address 1.1.1

**Purpose**

- Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end
or
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit

**Step 7**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>static-mac-address <em>(MAC address)</em></td>
<td>Configures the static MAC address to associate a remote MAC address with a pseudowire or any other bridge interface.</td>
</tr>
</tbody>
</table>

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#
static-mac-address 1.1.1

**Step 8**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>end or commit</td>
<td>Saves configuration changes.</td>
</tr>
</tbody>
</table>

- 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.
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2vpn RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group bridge group name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> bridge-domain bridge-domain name</td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> vfi (vfi name)</td>
<td>Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1 RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> neighbor (A.B.C.D) (pw-id value)</td>
<td>Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).</td>
</tr>
</tbody>
</table>
| **Example:** RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000 RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# | - 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

**Step 7**

- `pw-class {class name}`

_Example:_

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# pw-class canada
```

**Step 8**

- `end`
- `commit`

_Example:_

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end
```

### Purpose

- Configures the pseudowire class template name to use for the pseudowire.
- Saves configuration changes.
- When you issue the `end` command, the system prompts you to commit changes:

  - 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>l2vpn</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)#</code></td>
</tr>
<tr>
<td></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>bridge group</strong> <code>bridge group name</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco</code></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg)#</code></td>
</tr>
<tr>
<td></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>bridge-domain</strong> <code>bridge-domain name</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc</code></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</code></td>
</tr>
<tr>
<td></td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>vfi</strong> <code>{vfi name}</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi v1</code></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#</code></td>
</tr>
<tr>
<td></td>
<td>Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>neighbor</strong> <code>{A.B.C.D} (pw-id value)</code></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# neighbor 10.1.1.2 pw-id 1000</code></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)#</code></td>
</tr>
<tr>
<td></td>
<td>Adds an access pseudowire port to a bridge domain or a pseudowire to a bridge virtual forwarding interface (VFI).</td>
</tr>
<tr>
<td></td>
<td>- Use the <code>A.B.C.D</code> argument to specify the IP address of the cross-connect peer.</td>
</tr>
<tr>
<td></td>
<td>- Use the <code>pw-id</code> keyword to configure the pseudowire ID and ID value. The range is 1 to 4294967295.</td>
</tr>
</tbody>
</table>
### 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

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 7</td>
<td>mpls static label (local value) (remote value)</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# mpls static label local 800 remote 500</td>
</tr>
<tr>
<td>Step 8</td>
<td>end or commit</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# end or RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-pw)# commit</td>
</tr>
</tbody>
</table>

Configures the MPLS static labels and the static labels for the access pseudowire configuration. You can set the local and remote pseudowire labels.

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.
7. end
   or
   commit

8. show l2vpn bridge-domain [detail]

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><code>l2vpn</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><code>bridge group bridge group name</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td><code>bridge-domain bridge-domain name</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Configures virtual forwarding interface (VFI) parameters and enters L2VPN bridge group bridge domain VFI configuration mode.</td>
</tr>
<tr>
<td><code>vfi (vfi name)</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi vl</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Disables the virtual forwarding interface (VFI).</td>
</tr>
<tr>
<td><code>shutdown</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# shutdown</td>
<td></td>
</tr>
</tbody>
</table>
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-90
- Disabling the MAC Address Withdrawal, page VPC-92
- Configuring the MAC Address Limit, page VPC-95
- Configuring the MAC Address Aging, page VPC-97
- Disabling MAC Flush at the Bridge Port Level, page VPC-100

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Step 2 l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>Step 3 bridge group bridge group name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-12vpn)# bridge group cisco</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg)#</td>
<td></td>
</tr>
<tr>
<td>Step 4 bridge-domain bridge-domain name</td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg)# bridge-domain abc</td>
<td>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg-bd)#</td>
<td></td>
</tr>
<tr>
<td>Step 5 mac</td>
<td>Enters L2VPN bridge group bridge domain MAC configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg-bd)# mac</td>
<td>Enters L2VPN bridge group bridge domain MAC configuration mode.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg-bd-mac)#</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>learning disable</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# learning disable
```

<table>
<thead>
<tr>
<th>Step 7</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>end or commit</td>
<td>Saves configuration changes.</td>
</tr>
</tbody>
</table>

- 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.

<table>
<thead>
<tr>
<th>Step 8</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>show l2vpn bridge-domain [detail]</td>
<td>Displays the details that the MAC address source-based learning is disabled on the bridge.</td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RP0/CPU0:router# show l2vpn bridge-domain detail
```
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**
`configure` | Enters global configuration mode. |
| Example:
`RP/0/RP0/CPU0:router# configure` | |
| **Step 2**
`l2vpn` | Enters L2VPN configuration mode. |
| Example:
`RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#` | |
| **Step 3**
`bridge group bridge group name` | Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain. |
| Example:
`RP/0/RP0/CPU0:router(config-l2vpn)# bridge group
cisco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#` | |
| **Step 4**
`bridge-domain bridge-domain name` | Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode. |
| Example:
`RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain
abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#` | |
| **Step 5**
`mac` | Enters L2VPN bridge group bridge domain MAC configuration mode. |
| Example:
`RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#` | |
Implementing Virtual Private LAN Services

How to Implement Virtual Private LAN Services

VPC-93

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

The following sample output shows the MAC address withdrawal fields:

```
RP/0/RP0/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)

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 6 withdraw { access-pw disable</td>
<td>disable }</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# withdraw access-pw disable</td>
<td>Note Mac address withdrawal is generated when the access pseudowire is not operational.</td>
</tr>
<tr>
<td>Step 7 end or commit</td>
<td>Saves configuration changes.</td>
</tr>
</tbody>
</table>
| Example: RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# end or RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# commit | - 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] | 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. |
| Example: P/0/RP0/CPU0:router# show l2vpn bridge-domain detail | |

OL-24669-01

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

VPC-93
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 VFI:
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
<table>
<thead>
<tr>
<th>MPLS</th>
<th>Local</th>
<th>Remote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>3000</td>
<td>unknown</td>
</tr>
<tr>
<td>Group ID</td>
<td>0x0</td>
<td>0x0</td>
</tr>
<tr>
<td>Interface</td>
<td>siva/vfi</td>
<td>unknown</td>
</tr>
<tr>
<td>MTU</td>
<td>1500</td>
<td>unknown</td>
</tr>
<tr>
<td>Control word enabled</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>PW type</td>
<td>Ethernet</td>
<td>unknown</td>
</tr>
</tbody>
</table>

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.

**Note**

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

### SUMMARY STEPS

1. configure
2. l2vpn
3. bridge group bridge group name
4. bridge-domain bridge-domain name
5. 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CP0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CP0:router(config)# l2vpn RP/0/RP0/CP0:router(config-l2vpn)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> bridge group bridge group name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CP0:router(config-l2vpn)# bridge group cisco RP/0/RP0/CP0:router(config-l2vpn-bg)#</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step 4</th>
<th><code>bridge-domain bridge-domain name</code></th>
<th>Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 5</th>
<th><code>mac</code></th>
<th>Enters L2VPN bridge group bridge domain MAC configuration mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 6</th>
<th><code>limit</code></th>
<th>Sets the MAC address limit for action, maximum, and notification and enters L2VPN bridge group bridge domain MAC limit configuration mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# limit RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)#</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 7</th>
<th><code>maximum {value}</code></th>
<th>Configures the specified action when the number of MAC addresses learned on a bridge is reached.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# maximum 5000</td>
<td></td>
</tr>
</tbody>
</table>

| Step 8 | `action {flood | no-flood | shutdown}` | Configures the bridge behavior when the number of learned MAC addresses exceed the MAC limit configured. |
|--------|------------------------------------------|-----------------------------------------------------------------|
| Example| RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# action flood |

| Step 9 | `notification {both | none | trap}` | Specifies the type of notification that is sent when the number of learned MAC addresses exceeds the configured limit. |
|--------|--------------------------------------|-----------------------------------------------------------------|
| Example| RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)# notification both |
### 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}`

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 10</strong></td>
<td></td>
</tr>
<tr>
<td><code>end</code></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)#</td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)?</td>
</tr>
<tr>
<td><code>end</code></td>
<td>[cancel]:</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-limit)#</td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>– Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>– Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• Use the <code>commit</code> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td></td>
</tr>
<tr>
<td><code>show l2vpn bridge-domain [detail]</code></td>
<td>Displays the details about the MAC address limit.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router#</td>
<td></td>
</tr>
<tr>
<td><code>show l2vpn bridge-domain detail</code></td>
<td></td>
</tr>
</tbody>
</table>
## How to Implement Virtual Private LAN Services

9. **end**
   or
   **commit**

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure</td>
<td></td>
</tr>
</tbody>
</table>

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

| **Step 2** | Enters L2VPN configuration mode. |
| l2vpn      |         |

**Example:**
RP/0/RP0/CPU0:router(config)# l2vpn
RP/0/RP0/CPU0:router(config-l2vpn)#

| **Step 3** | Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain. |
| bridge group bridge group name |         |

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn)# bridge group cisco
RP/0/RP0/CPU0:router(config-l2vpn-bg)#

| **Step 4** | Establishes a bridge domain and enters L2VPN bridge group bridge domain configuration mode. |
| bridge-domain bridge-domain name |         |

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain abc
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)#

| **Step 5** | Enters L2VPN bridge group bridge domain MAC configuration mode. |
| mac |         |

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# mac
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)#

| **Step 6** | Enters the MAC aging configuration submode to set the aging parameters such as time and type. |
| aging |         |

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac)# aging
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#

| **Step 7** | Configures the maximum aging time. |
| time (seconds) |         |

**Example:**
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)# time 300

- 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.
<table>
<thead>
<tr>
<th>Step 8</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type {absolute</td>
<td>Configures the type for MAC address aging.</td>
</tr>
<tr>
<td></td>
<td>inactivity}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>type absolute</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 9</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>end</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td></td>
<td>or commit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-mac-aging)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>end</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or commit</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step 10</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>show l2vpn bridge-domain [detail]</td>
<td>Displays the details about the aging fields.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show l2vpn bridge-domain detail</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td>Step 2 l2vpn</td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-l2vpn)#</td>
</tr>
<tr>
<td>Step 3 bridge group bridge-group-name</td>
<td>Creates a bridge group so that it can contain bridge domains and then assigns network interfaces to the bridge domain.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-12vpn)# bridge group</td>
</tr>
<tr>
<td></td>
<td>cisco</td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg)#</td>
</tr>
<tr>
<td>Step 4 bridge-domain bridge-domain-name</td>
<td>Establishes a bridge domain and enters l2vpn bridge group bridge domain configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg)# bridge-domain</td>
</tr>
<tr>
<td></td>
<td>abc</td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-12vpn-bg-bd)#</td>
</tr>
</tbody>
</table>
Configuring VPLS with BGP Autodiscovery and Signaling

Perform this task to configure BGP-based autodiscovery and signaling.

**SUMMARY STEPS**

1. configure
2. l2vpn
3. bridge group bridge-group-name
4. bridge-domain bridge-domain-name
5. vfi {vfi-name}
6. `vpn-id` \textit{vpn-id}

7. `autodiscovery bgp`  

8. `rd` \textit{as-number:nn | ip-address:nn | auto}

9. `route-target` \textit{as-number:nn | ip-address:nn | export | import}

10. `route-target import` \textit{as-number:nn | ip-address:nn}

11. `route-target export` \textit{as-number:nn | ip-address:nn}

12. `signaling-protocol bgp`  

13. `ve-id` \textit{number}

14. `ve-range` \textit{number}

15. `commit`  

   or

   `end`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong>&lt;br&gt;<code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;<code>RP/0/RP0/CPU0:router# configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong>&lt;br&gt;<code>l2vpn</code></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;<code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong>&lt;br&gt;<code>bridge group</code> \textit{bridge-group-name}</td>
<td>Enters configuration mode for the named bridge group.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;<code>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group metroA</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong>&lt;br&gt;<code>bridge-domain</code> \textit{bridge-domain-name}</td>
<td>Enters configuration mode for the named bridge domain.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;<code>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain east</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong>&lt;br&gt;<code>vfi</code> \textit{vfi-name}</td>
<td>Enters virtual forwarding instance (VFI) configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;<code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong>&lt;br&gt;<code>vpn-id</code> \textit{vpn-id}</td>
<td>Specifies the identifier for the VPLS service. The VPN ID has to be globally unique within a PE router; that is the same VPN ID cannot exist in multiple VFIs on the same PE router. In addition, a VFI can have only one VPN ID.</td>
</tr>
<tr>
<td><strong>Example:</strong>&lt;br&gt;<code>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi)# vpn-id 100</code></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>7</td>
<td>autodiscovery bgp</td>
</tr>
<tr>
<td>8</td>
<td>rd {as-number::nn</td>
</tr>
<tr>
<td>9</td>
<td>route-target {as-number::nn</td>
</tr>
<tr>
<td>10</td>
<td>route-target import {as-number::nn</td>
</tr>
<tr>
<td>11</td>
<td>route-target export {as-number::nn</td>
</tr>
<tr>
<td>12</td>
<td>signaling-protocol bgp</td>
</tr>
<tr>
<td>13</td>
<td>ve-id (number)</td>
</tr>
</tbody>
</table>
## Configuring VPLS with BGP Autodiscovery and LDP Signaling

Perform this task to configure BGP-based Autodiscovery and signaling.

### SUMMARY STEPS

1. `configure`
2. `l2vpn`
3. `route-id`
4. `bridge group bridge-group-name`
5. `bridge-domain bridge-domain-name`
6. `vfi {vfi-name}
7. `autodiscovery bgp`
8. `vpn-id vpn-id`
9. `rd {as-number:nn | ip-address:nn | auto}`
10. `route-target {as-number:nn | ip-address:nn | export | import}`
11. `route-target import {as-number:nn | ip-address:nn}`

### Command or Action | Purpose
--- | ---
Step 14  
`ve-range (number)` | Overrides the minimum size of VPLS edge (VE) blocks. The default minimum size is 10. Any configured VE range must be higher than 10.

Example:
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-si g)# ve-range 40

Step 15  
`end`  
or  
`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.

Example:
RP0RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad - sig)# end
or
RP0RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad - sig)# commit

### Example:
```
RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# ve-range 40
```

```
RP0RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# end
or
RP0RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# commit
```
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters L2VPN configuration mode.</td>
</tr>
<tr>
<td>12vpn</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# 12vpn</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Specifies a unique Layer 2 (L2) router ID for the provider edge (PE) router.</td>
</tr>
<tr>
<td>router-id ip-address</td>
<td>The router ID must be configured for LDP signaling, and is used as the L2 router ID in the BGP NLRI, SAI (local L2 Router ID) and TAI (remote L2 Router ID). Any arbitrary value in the IPv4 address format is acceptable.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Each PE must have a unique L2 router ID. This CLI is optional, because a PE automatically generates a L2 router ID using the LDP router ID.</td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# router-id 1.1.1.1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enters configuration mode for the named bridge group.</td>
</tr>
<tr>
<td>bridge group bridge-group-name</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# bridge group metroA</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Enters configuration mode for the named bridge domain.</td>
</tr>
<tr>
<td>bridge-domain bridge-domain-name</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain east</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Enters virtual forwarding instance (VFI) configuration mode.</td>
</tr>
<tr>
<td>vfi (vfi-name)</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# vfi vfi-east</td>
<td></td>
</tr>
</tbody>
</table>
Implementing Virtual Private LAN Services

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><code>vpn-id</code> vpn-id</td>
<td>Specifies the identifier for the VPLS service. The VPN ID has to be globally unique within a PE router; that is the same VPN ID cannot exist in multiple VFIs on the same PE router. In addition, a VFI can have only one VPN ID.</td>
</tr>
<tr>
<td>8</td>
<td><code>autodiscovery bgp</code></td>
<td>Enters BGP autodiscovery configuration mode where all BGP autodiscovery parameters are configured. This command is not provisioned to BGP until the VPN ID and the signaling protocol is configured.</td>
</tr>
<tr>
<td>9</td>
<td><code>rd</code> `{as-number:nn</td>
<td>ip-address:nn</td>
</tr>
<tr>
<td>10</td>
<td><code>route-target</code> `{as-number:nn</td>
<td>ip-address:nn}`</td>
</tr>
<tr>
<td>11</td>
<td><code>route-target import</code> `{as-number:nn</td>
<td>ip-address:nn}`</td>
</tr>
<tr>
<td>12</td>
<td><code>route-target export</code> `{as-number:nn</td>
<td>ip-address:nn}`</td>
</tr>
<tr>
<td>13</td>
<td><code>signaling-protocol ldp</code></td>
<td>Enables BGP signaling, and enters the BGP signaling configuration submode where BGP signaling parameters are configured. This command is not provisioned to BGP until VE ID and VE ID range is configured.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpls-id (as-number:nn</td>
<td>ip-address:nn)</td>
</tr>
<tr>
<td>end</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

Example:
RP0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# vpls-id 10:20

Example:
RP0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# end
or
RP0/RP0/CPU0:router(config-l2vpn-bg-bd-vfi-ad-sig)# commit
PWHE Configuration Restrictions

These are the configuration restrictions for PWHE:

- Up to 4096 PWHE interfaces (a combination of pw-ether and pw-iw).
- Up to eight interface lists per peer.
- Up to eight 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 be configured on CRS only.
- Interface lists can accept POS, GigabitEthernet, TenGigabitEthernet, SRP, Bundle Ethernet and Bundle POS; 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router# configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>interface pw-ether id</strong></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/0/CPU0:router(config)# interface pw-ether &lt;id&gt;</td>
<td>Configures the PWHE interface and enters the interface configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>attach generic-interface-list interface_list_name</strong></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/0/CPU0:router(config-if)# attach generic-interface-list interfacelist1</td>
<td>Attaches the interface to a specified interface list.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>end</strong> or <strong>commit</strong></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-if)# end</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-if)# commit</td>
<td></td>
</tr>
<tr>
<td>- When you issue the <strong>end</strong> command, the system prompts you to commit changes:</td>
<td></td>
</tr>
<tr>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
<td></td>
</tr>
<tr>
<td>- Entering <strong>yes</strong> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
<td></td>
</tr>
<tr>
<td>- Entering <strong>no</strong> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>- Entering <strong>cancel</strong> leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>- Use the <strong>commit</strong> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
<td></td>
</tr>
</tbody>
</table>

### 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. end
   or
   commit
## Detailed Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

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

| Step 2 | `interface pw-ether id` | Configures the PWHE interface and enters the interface configuration mode. |

**Example:**

```
RP/0/0/CPU0:router(config)# interface pw-ether <id>
```

| Step 3 | `attach generic-interface-list interface_list_name` | Attaches the interface to a specified interface list. |

**Example:**

```
RP/0/0/CPU0:router(config-if)# attach generic-interface-list interfacelist1
```

| Step 4 | `l2overhead bytes` | Sets layer 2 overhead size. |

**Example:**

```
RP/0/0/CPU0:router(config-if)#l2overhead 20
```

| Step 5 | `load-interval seconds` | 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. |

**Example:**

```
RP/0/0/CPU0:router(config-if)#load-interval 90
```

| Step 6 | `dampening decay-life` | Configures state dampening on the given interface (in minutes). |

**Example:**

```
RP/0/0/CPU0:router(config-if)#dampening 10
```

| Step 7 | `logging events link-status` | Configures per interface logging. |

**Example:**

```
RP/0/0/CPU0:router(config-if)#logging events link-status
```

| Step 8 | `mac-address MAC address` | Sets the MAC address (xxxx.xxx.xxxx) on an interface. |

**Example:**

```
RP/0/0/CPU0:router(config-if)#mac-address aaaa.bbbb.cccc
```
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 9</td>
<td>mtu interface_MTU</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/0/CPU0:router(config-if)#mtu 128</td>
</tr>
<tr>
<td>Sets the MTU on an interface.</td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td>end</td>
</tr>
<tr>
<td>or</td>
<td>commit</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# end</td>
</tr>
<tr>
<td>or</td>
<td>RP/0/RSP0/CPU0:router(config-if)# commit</td>
</tr>
<tr>
<td>Saves configuration changes.</td>
<td></td>
</tr>
<tr>
<td>• When you issue the end command, the system prompts you to commit changes:</td>
<td></td>
</tr>
<tr>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
<td></td>
</tr>
<tr>
<td>– Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
<td></td>
</tr>
<tr>
<td>– Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>– Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>• Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
<td></td>
</tr>
</tbody>
</table>
## Detailed Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router# <code>configure</code> RP/0/RSP0/CPU0:router(config)#</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>l2vpn</code></td>
<td>Enters Layer 2 VPN configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# <code>l2vpn</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>xconnect group group-name</code></td>
<td>Configures a cross-connect group name using a free-format 32-character string.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-l2vpn)# <code>xconnect group MS-PW1</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>p2p xconnect-name</code></td>
<td>Enters P2P configuration submode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-l2vpn-xc)# <code>p2p ms-pw1</code></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>interface pw-ether id</code></td>
<td>Configures the PWHE interface.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# <code>interface pw-ether 100</code></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>neighbor A.B.C.D pw-id value</code></td>
<td>Configures a pseudowire for a cross-connect. The IP address is that of the corresponding PE node. The <code>pw-id</code> must match the <code>pw-id</code> of the PE node.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-l2vpn-xc-p2p)# <code>neighbor 10.165.200.25 pw-id 100</code></td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><code>pw-class class-name</code></td>
<td>Enters pseudowire class submode, allowing you to define a pseudowire class template.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-12vpn-xc-p 2p-pw)# pw-class dynamic_mpls</td>
<td></td>
</tr>
</tbody>
</table>
| 8    | `end` or `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. |
|      | **Example:** |
|      | RP/0/RSP0/CPU0:router(config-if)# end or RP/0/RSP0/CPU0:router(config-if)# commit | |
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-115
- Virtual Private LAN Services Configuration for Provider Edge-to-Customer Edge: Example, page VPC-116
- Configuring Backup Disable Delay: Example, page VPC-117
- Disabling MAC Flush: Examples, page VPC-117
- H-VPLS Configuration: Examples, page VPC-118
- Configuring VPLS with BGP Autodiscovery and Signaling: Example, page VPC-120
- Configuring Pseudowire Headend: Example, page VPC-125

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 1.1.1.1 255.255.255.255
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 2.2.2.2 255.255.255.255
commit

The following configuration example shows how to configure PE 3:
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
  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
```

```
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
end
```
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:

```plaintext
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:

```plaintext
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
```

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:

```plaintext
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:

```plaintext
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:

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

**H-VPLS Configuration: Examples**

This example shows how to configure hierarchical VPLS (H-VPLS). All examples in this section are based on the following topology where N-PE1 is the H-VPLS Node:

**VPLS with QinQ or QinAny: Example**

**Global Interface Configuration at N-PE1:**

```config
interface GigabitEthernet0/0/0/0
dot1q tunneling ethertype 0x9200
! interface GigabitEthernet0/0/0/1
dot1q tunneling ethertype 0x9100
! interface GigabitEthernet0/0/0.1 l2transport
dot1q vlan 20 21
! interface GigabitEthernet0/0/0/1.1 l2transport
dot1q vlan 10 any
```

**L2VPN Configuration at N-PE1:**

```config
l2vpn
  bridge group g1
  bridge-domain d1
    interface GigabitEthernet0/0/0/0.1
    ! interface GigabitEthernet0/0/0/1.1
```
Global Interface Configuration at N-PE2:

```conf
! vfi core-pws
 neighbor 6.6.6.6 pw-id 10
```

L2VPN Configuration at N-PE2:

```conf
l2vpn
bridge group g1
bridge-domain d1
interface GigabitEthernet0/6/0/0.1
interface GigabitEthernet0/6/0/1.1
vfi core-pws
 neighbor 5.5.5.5 pw-id 10
```

H-VPLS with Access-PWs: Example

**Router Configuration at U-PE1:**

```conf
l2vpn
pw-class vpls
  encapsulation mpls
  transport-mode ethernet
! xconnect group g1
  p2p p1
  interface GigabitEthernet0/1/1/0.1 --> Local AC
  neighbor 5.5.5.5 pw-id 100 --> Access PW to N-PE1
  pw-class vpls
interface GigabitEthernet0/1/1/0.1 l2transport
  dot1q vlan 1
```

**Router Configuration at U-PE2:**

```conf
l2vpn
pw-class vpls
  encapsulation mpls
  transport-mode ethernet
  mac-withdraw
! xconnect group g1
  p2p p1
  interface GigabitEthernet0/2/5/0.1 --> Local AC
  neighbor 5.5.5.5 pw-id 100 --> Access PW to N-PE1
```
pw-class vpls

interface GigabitEthernet0/2/5/0.1 l2transport
dot1q vlan 1

Router Configuration at N-PE1:

l2vpn
bridge group g1
bridge-domain d1
interface GigabitEthernet0/1/4/0.1 ? Local AC
neighbor 1.1.1.1 pw-id 100 --> Access PW to U-PE1
neighbor 2.2.2.2 pw-id 100 --> Access PW to U-PE2
!
vfi core1
neighbor 6.6.6.6 pw-id 100 --> Core PW to N-PE2

Router Configuration at N-PE2:

l2vpn
bridge group g1
bridge-domain d1
interface GigabitEthernet0/1/4/0.1 --> Local AC

vfi core1
neighbor 5.5.5.5 pw-id 100 --> Core PW to N-PE1

Configuring VPLS with BGP Autodiscovery and Signaling: Example

This section contains these configuration examples:

- LDP and BGP Configuration
- Minimum L2VPN Configuration for BGP Autodiscovery with BGP Signaling
- VPLS with BGP Autodiscovery and BGP Signaling
- Minimum Configuration for BGP Autodiscovery with LDP Signaling
- VPLS with BGP Autodiscovery and LDP Signaling

LDP and BGP Configuration

Figure 17 illustrates an example of LDP and BGP configuration.
Implementing Virtual Private LAN Services

Configuration Examples for Virtual Private LAN Services

VPC-121

Cisco IOS XR Virtual Privat e Network Configuration Guide for the Cisco CRS Router

OL-24669-01

Configuration at PE1:

interface Loopback0
   ipv4 address 1.1.1.100 255.255.255.255
!
interface Loopback1
   ipv4 address 1.1.1.10 255.255.255.255
!
mls ldp
   router-id 1.1.1.1
   interface GigabitEthernt0/1/0/0
!
rout er bgp 120
   address-family l2vpn vpls-vpws
!
neighbor 2.2.2.20
   remote-as 120
   update-source Loopback1
   address-family l2vpn vpls-vpws
   signaling bgp disable

Configuration at PE2:

interface Loopback0
   ipv4 address 2.2.2.200 255.255.255.255
!
interface Loopback1
   ipv4 address 2.2.2.20 255.255.255.255
!
mls ldp
   router-id 2.2.2.2
   interface GigabitEthernt0/1/0/0
!
rout er bgp 120
   address-family l2vpn vpls-vpws
!
neighbor 1.1.1.10
   remote-as 120
   update-source Loopback1
   address-family l2vpn vpls-vpws

Minimum L2VPN Configuration for BGP Autodiscovery with BGP Signaling

This example illustrates the minimum L2VPN configuration required for BGP Autodiscovery with BGP Signaling, where any parameter that has a default value is not configured.

(config)# l2vpn
(config-l2vpn)# bridge group {bridge group name}
(config-l2vpn-bg)# bridge-domain {bridge domain name}
(config-l2vpn-bg-bd)# vfi {vfi name}
(config-l2vpn-bg-bd-vfi)# vpn-id 10
(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
(config-l2vpn-bg-bd-vfi-ad)# rd auto
(config-l2vpn-bg-bd-vfi-ad)# route-target 1.1.1.1:100
(config-l2vpn-bg-bd-vfi-ad)# signaling-protocol bgp
(config-l2vpn-bg-bd-vfi-ad-sig)# ve-id 1
(config-l2vpn-bg-bd-vfi-ad-sig)# commit

VPLS with BGP Autodiscovery and BGP Signaling

Figure 18 illustrates an example of configuring VPLS with BGP autodiscovery (AD) and BGP Signaling.
Figure 18: VPLS with BGP autodiscovery and BGP signaling

Configuration at PE1:
```conf
l2vpn
bridge group gr1
bridge-domain bd1
    interface GigabitEthernet0/1/0/1.1
    vfi vf1
! AD independent VFI attributes
    vpn-id 100
! Auto-discovery attributes
    autodiscovery bgp
    rd auto
    route-target 2.2.2.2:100
! Signaling attributes
    signaling-protocol bgp
    ve-id 3
```

Configuration at PE2:
```conf
l2vpn
bridge group gr1
bridge-domain bd1
    interface GigabitEthernet0/1/0/2.1
    vfi vf1
! AD independent VFI attributes
    vpn-id 100
! Auto-discovery attributes
    autodiscovery bgp
    rd auto
    route-target 2.2.2.2:100
! Signaling attributes
    signaling-protocol bgp
    ve-id 5
```

This is an example of NLRI for VPLS with BGP AD and signaling:

Discovery attributes
NLRI sent at PE1:
- Length = 19
- Router Distinguisher = 3.3.3.3:32770
- VE ID = 3
- VE Block Offset = 1
- VE Block Size = 10
- Label Base = 16015
NLRI sent at PE2:
Length = 19
Router Distinguisher = 1.1.1.1:32775
VE ID = 5
VE Block Offset = 1
VE Block Size = 10
Label Base = 16120

Minimum Configuration for BGP Autodiscovery with LDP Signaling

This example illustrates the minimum L2VPN configuration required for BGP Autodiscovery with LDP Signaling, where any parameter that has a default value is not configured.

```
(config)# l2vpn
(config-l2vpn)# bridge group {bridge group name}
(config-l2vpn-bg)# bridge-domain {bridge domain name}
(config-l2vpn-bg-bd)# vfi {vfi name}
(config-l2vpn-bg-bd-vfi)# autodiscovery bgp
(config-l2vpn-bg-bd-vfi-ad)# vpn-id 10
(config-l2vpn-bg-bd-vfi-ad)# rd auto
(config-l2vpn-bg-bd-vfi-ad)# route-target 1.1.1.1:100
(config-l2vpn-bg-bd-vfi-ad)# commit
```

VPLS with BGP Autodiscovery and LDP Signaling

Figure 19 illustrates an example of configuring VPLS with BGP autodiscovery (AD) and LDP Signaling.

Figure 19  VPLS with BGP autodiscovery and LDP signaling

Configuration at PE1:
```
l2vpn
    router-id 10.10.10.10
    bridge group bg1
    bridge-domain bd1
    vfi vf1
    vpn-id 100
    autodiscovery bgp
    rd 1:100
    router-target 12:12
```

Configuration at PE2:
```
l2vpn
    router-id 20.20.20.20
    bridge group bg1
    bridge-domain bd1
    vfi vf1
    vpn-id 100
    autodiscovery bgp
    rd 2:200
```
Implementing Virtual Private LAN Services

Configuration Examples for Virtual Private LAN Services

VPC-124

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

OL-24669-01

router-target 12:12
signaling-protocol ldp
vpls-id 120:100

Discovery and Signaling Attributes

Configuration at PE1:
LDP Router ID - 1.1.1.1
BGP Router ID - 1.1.1.100
Peer Address - 1.1.1.10
L2VPN Router ID - 10.10.10.10
Route Distinguisher - 1:100

Common Configuration between PE1 and PE2:
ASN - 120
VPN ID - 100
VPLS ID - 120:100
Route Target - 12:12

Configuration at PE2:
LDP Router ID - 2.2.2.2
BGP Router ID - 2.2.2.200
Peer Address - 2.2.2.20
L2VPN Router ID - 20.20.20.20
Route Distinguisher - 2:200

Discovery Attributes

NLRI sent at PE1:
Source Address - 1.1.1.10
Destination Address - 2.2.2.20
Length - 14
Route Distinguisher - 1:100
L2VPN Router ID - 10.10.10.10
VPLS ID - 120:100
Route Target - 12:12

NLRI sent at PE2:
Source Address - 2.2.2.20
Destination Address - 1.1.1.10
Length - 14
Route Distinguisher - 2:200
L2VPN Router ID - 20.20.20.20
VPLS ID - 120:100
Route Target - 12:12
Configuring Pseudowire Headend: Example

This section provides an example of pseudowire headend configuration.

Figure 20    PWHE Configuration Example

Consider the topology in Figure 20.

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
```
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 pin down 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 ill
    interface gig0/1/0/0
    interface gig0/2/0/0

generic-interface-list ill2
    interface gig0/1/0/1
    interface gig0/2/0/1

interface pw-ether1
    ipv4 address 10.1.1.2/24
    attach generic-interface-list ill
```
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

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

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XR L2VPN command reference document</td>
<td>MPLS Virtual Private Network Commands on Cisco IOS XR Software module in Cisco IOS XR MPLS Command Reference</td>
</tr>
<tr>
<td>MPLS VPLS-related commands</td>
<td>MPLS Virtual Private LAN Services Commands on Cisco IOS XR Software module in Cisco IOS XR MPLS Command Reference</td>
</tr>
<tr>
<td>MPLS Layer 2 VPNs</td>
<td>Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software module in Cisco IOS XR MPLS Configuration Guide</td>
</tr>
<tr>
<td>MPLS VPNs over IP Tunnels</td>
<td>MPLS VPNs over IP Tunnels on Cisco IOS XR Software module in Cisco IOS XR MPLS Configuration Guide</td>
</tr>
<tr>
<td>Cisco CRS router getting started material</td>
<td>Cisco IOS XR Getting Started Guide</td>
</tr>
<tr>
<td>Information about user groups and task IDs</td>
<td>Configuring AAA Services on Cisco IOS XR Software module of Cisco IOS XR System Security Configuration Guide</td>
</tr>
</tbody>
</table>

Standards

<table>
<thead>
<tr>
<th>Standards1</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

1. Not all supported standards are listed.
MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 3931</td>
<td>Layer Two Tunneling Protocol - Version 3 (L2TPv3)</td>
</tr>
<tr>
<td>RFC 4447</td>
<td>Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP), April 2006</td>
</tr>
<tr>
<td>RFC 4448</td>
<td>Encapsulation Methods for Transport of Ethernet over MPLS Networks, April 2006</td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.7.0</td>
<td>This feature was introduced. Support was added for Inter-AS 6PE.</td>
</tr>
<tr>
<td>Release 4.1.0</td>
<td>Support for the Open Shortest Path First version 3 (OSPFv3) IPv6 VPN Provider Edge (6VPE) feature was added.</td>
</tr>
</tbody>
</table>

Contents

- Prerequisites for Implementing 6PE, page VPC-128
- Information About 6PE, page VPC-128
- How to Implement 6PE, page VPC-131
- Configuration Examples for 6PE, page VPC-136
- Additional References, page VPC-137
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-128
- Benefits of 6PE, page VPC-128
- Deploying IPv6 over MPLS Backbones, page VPC-129
- IPv6 on the Provider Edge and Customer Edge Routers, page VPC-129
- IPv6 Provider Edge Multipath, page VPC-130
- OSPFv3 6VPE, page VPC-130

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.
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.

OSPFv3 6VPE

The Open Shortest Path First version 3 (OSPFv3) IPv6 VPN Provider Edge (6VPE) feature adds VPN routing and forwarding (VRF) and provider edge-to-customer edge (PE-CE) routing support to Cisco IOS XR OSPFv3 implementation. This feature allows:

- Multiple VRF support per OSPFv3 routing process
- OSPFV3 PE-CE extensions
Multiple VRF Support

OSPFv3 supports multiple VRFs in a single routing process that allows scaling to tens and hundreds of VRFs without consuming too much route processor (RP) resources.

Multiple OSPFv3 processes can be configured on a single router. In large-scale VRF deployments, this allows partition VRF processing across multiple RPs. It is also used to isolate default routing table or high impact VRFs from the regular VRFs. It is recommended to use a single process for all the VRFs. If needed, a second OSPFv3 process must be configured for IPv6 routing.

Note
The maximum of four OSPFv3 processes are supported.

OSPFv3 PE-CE Extensions

IPv6 protocol is being vastly deployed in today’s customer networks. Service Providers (SPs) need to be able to offer Virtual Private Network (VPN) services to their customers for supporting IPv6 protocol, in addition to the already offered VPN services for IPv4 protocol.

In order to support IPv6, routing protocols require additional extensions for operating in the VPN environment. Extensions to OSPFv3 are required in order for OSPFv3 to operate at the PE-CE links.

VRF Lite

VRF lite feature enables VRF deployment without BGP or MPLS based backbone. In VRF lite, the PE routers are directly connected using VRF interfaces. For OSPFv3, the following needs to operate differently in the VRF lite scenario, as opposed to the deployment with BGP or MPLS backbone:

- DN bit processing—In VRF lite environment, the DN bit processing is disabled.
- ABR status—In VRF context (except default VRF), OSPFv3 router is automatically set as an ABR, regardless to it’s connectivity to area 0. This automatic ABR status setting is disabled in the VRF lite environment.

Note
To enable VRF Lite, issue the capability vrf-lite command in the OSPFv3 VRF configuration submode.

How to Implement 6PE

This section includes the following implementation procedure:

- Configuring 6PE, page VPC-131
- Configuring OSPFv3 as the Routing Protocol Between the PE and CE Routers, page VPC-133

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.
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp as-number</td>
<td>Enters the number that identifies the autonomous system (AS) in which the router resides.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router bgp 1</td>
<td>Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.</td>
</tr>
<tr>
<td>Step 3 neighbor ip-address</td>
<td>Enters neighbor configuration mode for configuring Border Gateway Protocol (BGP) routing sessions.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp)# neighbor 1.1.1.1</td>
<td></td>
</tr>
<tr>
<td>Step 4 address-family ipv6 labeled-unicast</td>
<td>Specifies IPv6 labeled-unicast address prefixes.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv6 labeled-unicast</td>
<td>Note This option is also available in IPv6 neighbor configuration mode and VRF neighbor configuration mode.</td>
</tr>
<tr>
<td>Step 5 exit</td>
<td>Exits BGP address-family submode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# exit</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring OSPFv3 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 version 3 (OSPFv3).

**SUMMARY STEPS**

1. `configure`  
2. `router ospfv3` *process-name*  
3. `vrf` *vrf-name*  
4. `capability vrf-lite`
5. `router-id {router-id | type interface-path-id}`
6. `domain-id type {0005 | 0105 | 0205 | 8005} value domain-id`
7. `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]`  
or  
`redistribute rip [metric metric-value] [metric-type {1 | 2}] [route-policy policy-name] [tag tag-value]`
8. `area area-id`
9. `interface type interface-path-id`
10. `end`
   or  
`commit`

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>configure</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters OSPF configuration mode allowing you to configure the OSPF routing process.</td>
</tr>
<tr>
<td>router ospf process-name</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router ospf 109</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.</td>
</tr>
<tr>
<td>vrf vrf-name</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ospf)# vrf vrf_1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Enables VRF Lite feature.</td>
</tr>
<tr>
<td>capability vrf-lite</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ospf-vrf)# capability vrf-lite</td>
<td></td>
</tr>
</tbody>
</table>
Step 5

**Command or Action**

```
router-id (router-id | type interface-path-id)
```

**Example:**

```
RP/0/RP0/CPU0:router(config-ospf-vrf)#
router-id 172.20.10.10
```

**Purpose**

Configures the router ID for the VRF.

**Note**

Router ID configuration is required for each VRF.

Step 6

**Command or Action**

```
domain-id type (0005 | 0105 | 0205 | 8005) value domain-id
```

**Example:**

```
RP/0/RP0/CPU0:router(config-ospf-vrf)#
domain-id type 0005 value CAFB00112233
```

**Purpose**

Specifies the domain ID.

Step 7

**Command or Action**

```
redistribute bgp process-id [metric metric-value] [route-policy policy-name] [tag tag-value] or
redistribute connected [metric metric-value] [route-policy policy-name] [tag tag-value] or
redistribute ospf process-id [match [external [1 | 2] | internal | nssa-external [1 | 2]]] [metric metric-value] [route-policy policy-name] [tag tag-value] or
redistribute static [metric metric-value] [route-policy policy-name] [tag tag-value] or
redistribute eigrp process-id [match [external [1 | 2] | internal | nssa-external [1 | 2]]] [metric metric-value] [route-policy policy-name] [tag tag-value] or
redistribute rip [metric metric-value] [route-policy policy-name] [tag tag-value]
```

**Example:**

```
RP/0/RP0/CPU0:router(config-ospf-vrf)#
redistribute connected
```

**Purpose**

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 8

**Command or Action**

```
area area-id
```

**Example:**

```
RP/0/RP0/CPU0:router(config-ospf-vrf)#
area 0
```

**Purpose**

Configures the OSPF area as area 0.
### Configuration Examples for 6PE

This section includes the following configuration example:
- Configuring 6PE on a PE Router: Example, page VPC-136
- Configuring OSPFv3 6VPE: Example, page VPC-137

### Configuring 6PE on a PE Router: Example

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

```conf
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
```

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 9</strong></td>
<td>Associates interface GigabitEthernet 0/3/0/0 with area 0.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/3/0/0</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-ospf-vrf-ar-if)# end or commit</td>
</tr>
<tr>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
<td></td>
</tr>
<tr>
<td>– Entering <strong>yes</strong> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
<td></td>
</tr>
<tr>
<td>– Entering <strong>no</strong> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>– Entering <strong>cancel</strong> leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>• Use the <strong>commit</strong> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
<td></td>
</tr>
</tbody>
</table>
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

**Configuring OSPFv3 6VPE: Example**

This example shows you how to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Open Shortest Path First version 3 (OSPFv3):

```plaintext
router ospfv3 0
    vrf V1
        router-id 100.0.0.2
        domain-id type 0005 value CAFE00112233
        domain-id secondary type 0105 value beef00000001
        domain-id secondary type 0205 value beef00000002
        capability vrf-lite
        redistribute bgp 1
        area 0
            interface POS0/3/0/1
        vrf V2
            router-id 200.0.0.2
            capability vrf-lite
            area 1
                interface POS0/3/0/2
```

**Additional References**

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

**Related Document**

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XR L2VPN command reference document</td>
<td>MPLS Virtual Private Network Commands on Cisco IOS XR Software module in Cisco IOS XR MPLS Command Reference</td>
</tr>
<tr>
<td>Cisco CRS router getting started material</td>
<td>Cisco IOS XR Getting Started Guide</td>
</tr>
<tr>
<td>Information about user groups and task IDs</td>
<td>Configuring AAA Services on Cisco IOS XR Software module in Cisco IOS XR System Security Configuration Guide</td>
</tr>
</tbody>
</table>
Standards

<table>
<thead>
<tr>
<th>Standards¹</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>—</td>
</tr>
</tbody>
</table>

1. Not all supported standards are listed.

MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.9.0</td>
<td>This feature was introduced.</td>
</tr>
</tbody>
</table>

Contents

- Prerequisites for Layer 2 Tunnel Protocol Version 3, page VPC-139
- Information About Layer 2 Tunnel Protocol Version 3, page VPC-140
- How to Implement Layer 2 Tunnel Protocol Version 3, page VPC-147
- Configuration Examples for Layer 2 Tunnel Protocol Version 3, page VPC-166
- Additional References, page VPC-167

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, page VPC-140
- L2TPv3 Benefits, page VPC-141
- L2TPv3 Features, page VPC-141

**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 and 802.1q (VLAN) using the sessions described in the following sections:

- Static L2TPv3 Sessions, page VPC-142
- Dynamic L2TPv3 Sessions, page VPC-142

L2TPv3 also supports:

- Local Switching, page VPC-142
- Local Switching: Quality of Service, page VPC-144
- L2TPv3 Pseudowire Switching, page VPC-144
- L2TPv3 Pseudowire Manager, page VPC-144
- IP Packet Fragmentation, page VPC-144
- L2TPv3 Type of Service Marking, page VPC-145
- Keepalive, page VPC-145
- Maximum Transmission Unit Handling, page VPC-145
- 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
- IPSec Mapping to L2TPv3, page VPC-146
- Like-to-Like Pseudowires, page VPC-146
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.

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.

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:

- 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. 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.
Figure 23  Local Switching Operation

![Diagram showing local switching operation with interconnected interfaces and tunneled LANs.

- CE R3
- CE R4
- PE R1
- L2TPv3 tunneled LAN
- LAN1
- LAN2

Legend:
- Xconnected interface
- int1, int2, int4
- e1

Description:
- CE R3 and CE R4 are connected via an Xconnected interface.
- PE R1 connects CE R3 and CE R4 to LAN1 and LAN2.
- L2TPv3 tunneled LANs are used for communication between LAN1 and LAN2.]

---

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

OL-24669-01
Local Switching: Quality of Service

The following quality of service (QoS) requirements apply to 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 L3 VLANs or L2 VLAN 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 StopCCCN 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.

IP Security Mapping to L2 Tunneling Protocol, Version 3

This feature is supported only on the Cisco IPSec VPN SPA.

The L2TPv3 is a protocol that is used to tunnel a variety of payload types over IP networks. IP security (IPSec) provides an additional level of protection at a service PE router than relying on access control list (ACL) filters. L2TPv3 tunnels are also secured by using IPSec, as specified in RFC3931.

You can secure L2TPv3 tunnels by using IPSec, which provides authentication, privacy protection, integrity checking, and replay protection. When using IPSec, the tunnel head and the tunnel tail can be treated as the endpoints of an SA. A single IP address of the tunnel head is used as the source IP address, and a single IP address of the tunnel tail is used as the destination IP address.
The following scenarios are described to have L2TPv3 work with IPSec:
- IPSec Mapping to L2TPv3, page VPC-146
- IPSec over L2TPv3, page VPC-146

**IPSec Mapping to L2TPv3**

A CE 1 router sends an IPSec packet to a PE1 router. The PE1 router sends an IPSec packet to the Cisco IPSec VPN SPA by routing the look up for the front door virtual routing and forwarding (FVRF) in the service-ipsec interface. The Cisco IPSec VPN SPA can decapsulate an IPSec packet to obtain a clear IP packet, and perform a routing look up for the inside virtual routing and forwarding (IVRF) in the service-ipsec interface.

**IPSec over L2TPv3**

If the packet arrives at PE1 outside of a virtual routing and forwarding (VRF), for example, the global table, the packet is forwarded to the PE2 according to the global FIB in PE1. This is normal for IP switching until the packet arrives at PE2 with no encapsulation at any point.

**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 following features describe the pseudowire connection:
- PPP, page VPC-146

**PPP**

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 connection.

---

**Note**

In an MPLS network, pseudowires are carried inside an LSP tunnel.
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-147 (required)
- Configuring L2TP Control-Channel Parameters, page VPC-149 (required)
- Configuring L2TPv3 Pseudowires, page VPC-159 (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. protocol l2tpv3 class class name
6. ipv4 source ip-address
7. transport mode {ethernet | vlan}
8. end
    or
    commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>l2vpn</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config)# l2vpn</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>pw-class class name</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-l2vpn)# pw-class wkg</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td></td>
</tr>
<tr>
<td><code>encapsulation l2tpv3</code></td>
<td>Configures pseudowire encapsulation to the Layer 2 Tunnel Protocol.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2tp-pwc)# encapsulation l2tpv3</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td></td>
</tr>
<tr>
<td><code>protocol l2tpv3 class class name</code></td>
<td>Configures the L2TPv3 dynamic pseudowire signaling protocol to be used to manage the pseudowires created.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# protocol l2tpv3 class Class_l2tp_01</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td></td>
</tr>
<tr>
<td><code>ipv4 source ip-address</code></td>
<td>Configures the local source IPv4 address.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# ipv4 source 126.10.1.55</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td></td>
</tr>
<tr>
<td>`transport-mode {ethernet</td>
<td>vlan}`</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# transport-mode ethernet</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td></td>
</tr>
<tr>
<td><code>end</code></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# end</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-l2tp-pwc-encap-l2tpv3)# commit</td>
<td></td>
</tr>
</tbody>
</table>

**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.

- 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-149
- Configuring L2TPv3 Control-Channel Authentication Parameters, page VPC-150
- Configuring L2TP Control-Channel Maintenance Parameters, page VPC-159

**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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Rp/0/Rp0/Cpu0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>l2tp-class l2tp-class-name</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Rp/0/Rp0/Cpu0:router(config)# l2tp-class cisco</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>receive-window size</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Rp/0/Rp0/Cpu0:router(config-l2tp-class)# receive-window 30</td>
</tr>
</tbody>
</table>
| **Step 4**        | **retransmit (initial retries initial-retries | retries retries | timeout (max | min) timeout)**<br>**Example:**<br>Rp/0/Rp0/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**<br>**Example:**<br>Rp/0/Rp0/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-151)
- L2TPv3 Control Message Hashing (see Configuring L2TPv3 Control Message Hashing, page VPC-152)

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-154,) 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> l2tp-class word</td>
<td>Specifies the L2TP class name and enters L2TP class configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2tp-class class1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> authentication</td>
<td>Enables authentication for the control-channel between PE routers.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2tp-class)# authentication</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> password {0</td>
<td>7} password</td>
</tr>
</tbody>
</table>
| **Example:** RP/0/RP0/CPU0:router(config-l2tp-class)# password 7 cisco | - [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 | Specifies a hostname used to identify the router during L2TP control-channel authentication. |
| **Example:** RP/0/RP0/CPU0:router(config-l2tp-class)# hostname yb2 | - 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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Specifies the L2TP class name and enters L2TP class configuration mode.</td>
</tr>
<tr>
<td><code>l2tp-class word</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config)# l2tp-class class1</td>
</tr>
</tbody>
</table>
### 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-152).

**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.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| digest (check disable | hash (MD5 | SHA1)) | secret (0 | 7) password) | 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.  
    • 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. |
| hidden | Enables AVP hiding when sending control messages to an L2TPv3 peer. |

Example:

```
RP/0/RP0/CPU0:router(config-l2tp-class)# digest secret cisco hash sha
```

Example:

```
RP/0/RP0/CPU0:router(config-l2tp-class)# hidden
```
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td>Enters global configuration mode.</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>l2tp-class word</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config)# l2tp-class class1</td>
</tr>
<tr>
<td>Specifies the L2TP class name and enters L2TP class configuration mode.</td>
<td></td>
</tr>
</tbody>
</table>
Step 3

```
digest {check disable | hash {MD5 | SHA1}} | secret {0 | 7} password
```

Example:
```
RP/0/RP0/CPU0:router(config-l2tp-class)# digest
secret cisco hash sha
```

**Purpose**
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}`—Specifications 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
```

**Purpose**
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

#### Step 5
**show l2tp tunnel brief**

**Example:**
RP/0/RP0/CPU0:router# show l2tun tunnel brief

**Purpose:**
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/RP0/CPU0:router# configure

**Purpose:**
Enters global configuration mode.

#### Step 7
**l2tp-class word**

**Example:**
RP/0/RP0/CPU0:router(config)# l2tp-class class1

**Purpose:**
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/RP0/CPU0:router(config-l2tp-class)# no digest secret cisco hash sha1

**Purpose:**
Disables L2TPv3 control-channel authentication or integrity checking.
Step 9

**Command or Action**

- `end`
- `commit`

**Example:**

```
RP/0/RP0/CPU0:router(config-l2tp-class)# end
```

```
RP/0/RP0/CPU0:router(config-l2tp-class)# commit
```

**Purpose**

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

**Command or Action**

- `show l2tp tunnel brief`

**Example:**

```
RP/0/RP0/CPU0:router# show l2tun tunnel brief
```

**Purpose**

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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 2</strong> <code>l2tp-class word</code></td>
<td>Specifies the L2TP class name and enters L2TP class configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**
```
RP/0/RP0/CPU0:router(config)# l2tp-class class1
```

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 3</strong> <code>hello interval</code></td>
<td>Specifies the exchange interval (in seconds) used between L2TP hello packets.</td>
</tr>
</tbody>
</table>

**Example:**
```
RP/0/RP0/CPU0:router(config-l2tp-class)# hello 100
```

- 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-159
- Configuring a Static L2TPv3 Pseudowire, page VPC-162

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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <strong>configure</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <strong>l2vpn</strong></td>
<td>Enter L2VPN configure submode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <strong>xconnect group</strong> <em>name</em></td>
<td>Enter a name for the cross-connect group.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group grp_01</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <strong>p2p</strong> <em>name</em></td>
<td>Enters p2p configuration submode to configure point-to-point cross-connects.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> <strong>neighbor</strong> <em>ip-address</em> <strong>pw-id</strong> <em>number</em></td>
<td>Configures a pseudowire for a cross-connect.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p)# neighbor 10.1.1.1 pw-id 665</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> <strong>pw-class</strong> <em>pw-class-name</em></td>
<td>Enters pseudowire class submode to define a name for the cross-connect.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# pw-class atom</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>end or commit</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# end or
RP/0/RP0/CPU0:router(config-l2vpn-xc-p2p-pw)# commit
```

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.

<table>
<thead>
<tr>
<th>Step 8</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>pw-class pw-class-name</code></td>
<td>Enters pseudowire class submode to define a pseudowire class template.</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn)# pw-class class100
```

<table>
<thead>
<tr>
<th>Step 9</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>encapsulation l2tpv3</code></td>
<td>Configures L2TPv3 pseudowire encapsulation.</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3
```
## Configuring a Static 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

### Command or Action

**Step 10**

```plaintext
protocol l2tpv3 class class name
```

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# protocol l2tpv3 class wkg
```

**Step 11**

```plaintext
end
```

or

```plaintext
commit
```

**Example:**

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# end
```

or

```
RP/0/RP0/CPU0:router(config-l2vpn-pwc-encap-l2tpv3)# commit
```

### Purpose

- Configures the dynamic pseudowire signaling protocol.
- 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

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><code>l2vpn</code></td>
<td>Enter L2VPN configure submode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config)# l2vpn</code></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>xconnect group name</code></td>
<td>Enter a name for the cross-connect group.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn)# xconnect group customer_X</code></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><code>p2p name</code></td>
<td>Enters p2p configuration submode to configure point-to-point cross-connects.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn-xc)# p2p AC1_to_PW1</code></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>neighbor ip-address pw-id number</code></td>
<td>Configures a pseudowire for a cross-connect.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn- xc-p2p)# neighbor 10.1.1.1 pw-id 666</code></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>l2tp static local session {session-id}</code></td>
<td>Configures a L2TP pseudowire static session ID.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn- xc-p2p-pw)# l2tp static local session 147</code></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>`l2tp static local cookie size {0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-l2vpn- xc-p2p-pw)# l2tp static local cookie size 4 value 0XA</code></td>
<td></td>
</tr>
</tbody>
</table>

---

**DETAILED STEPS**

1. **l2vpn**
2. `pw-class name`
3. `encapsulation l2tpv3`
4. `ipv4 source ip-address`
5. `end`
   or `commit`
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 8</td>
<td><code>l2tp static remote session {session-id}</code></td>
<td>Configures a L2TP pseudowire remote session ID.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RPO/CPU0:router(config-12vpn-xc-p2p-pw)# l2tp static remote session 123</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>`l2tp static remote cookie size (0</td>
<td>4</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RPO/CPU0:router(config-12vpn-xc-p2p-pw)# l2tp static remote cookie size 8 value 0x456 0xFFB</td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td><code>pw-class name</code></td>
<td>Enters pseudowire class submode to define a pseudowire class template.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RPO/CPU0:router(config-12vpn-xc-p2p-pw)# pw-class atom</td>
<td></td>
</tr>
<tr>
<td>Step 11</td>
<td><code>end</code> or <code>commit</code></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RPO/CPU0:router(config-12vpn-xc-p2p-pw)# end or RP/0/RPO/CPU0:router(config-12vpn-xc-p2p-pw)# commit</td>
<td></td>
</tr>
<tr>
<td>Step 12</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RPO/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>Step 13</td>
<td><code>l2vpn</code></td>
<td>Enter L2VPN configure submode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RPO/CPU0:router(config)# l2vpn</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

**Step 14**  
`pw-class name`  
**Example:**  
```
RP/0/RP0/CPU0:router(config-l2vpn)# pw-class class100
```
- Enters pseudowire class submode to define a pseudowire class template.

**Step 15**  
`encapsulation l2tpv3`  
**Example:**  
```
RP/0/RP0/CPU0:router(config-l2vpn-pwc)# encapsulation l2tpv3
```
- Configures L2TPv3 pseudowire encapsulation.

**Step 16**  
`ipv4 source ip-address`  
**Example:**  
```
RP/0/RP0/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/RP0/CPU0:router(config-l2vpn-pwc)# end
```
- or  
```
RP/0/RP0/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.
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, page VPC-166
- Configuring a Pseudowire Class: Example, page VPC-166
- Configuring L2TPv3 Control Channel Parameters: Example, page VPC-166
- Configuring an Interface for Layer 2 Transport Mode: Example, page VPC-167

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.

```
close
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
digest secret cisco hash MD5
```

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}]
dfbbit set
tos {reflect|value {value}}
ttl {1-255}
pmtu max {68-65535}
ipv4 source {ipv4_address}
```

Configuring L2TPv3 Control Channel Parameters: Example

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

```
configure
l2tp-class FR-12tp
authentication
digest secret cisco hash MD5
digest secret cisco hash MD5
```

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router
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

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPLS VPN-related commands</td>
<td>MPLS Virtual Private Network Commands on Cisco IOS XR Software</td>
</tr>
<tr>
<td></td>
<td>module in Cisco IOS XR MPLS Command Reference</td>
</tr>
<tr>
<td>MPLS Layer 2 VPNs</td>
<td>Implementing MPLS Layer 2 VPNs on Cisco IOS XR Software module</td>
</tr>
<tr>
<td></td>
<td>in Cisco IOS XR MPLS Configuration Guide</td>
</tr>
<tr>
<td>MPLS Layer 3 VPNs</td>
<td>Implementing MPLS Layer 3 VPNs on Cisco IOS XR Software module</td>
</tr>
<tr>
<td></td>
<td>in Cisco IOS XR MPLS Configuration Guide</td>
</tr>
<tr>
<td>MPLS VPNs over IP Tunnels</td>
<td>MPLS VPNs over IP Tunnels on Cisco IOS XR Software module</td>
</tr>
<tr>
<td></td>
<td>in Cisco IOS XR MPLS Configuration Guide</td>
</tr>
<tr>
<td>Cisco CRS router getting started material</td>
<td>Cisco IOS XR Getting Started Guide</td>
</tr>
<tr>
<td>Information about user groups and task IDs</td>
<td>Configuring AAA Services on Cisco IOS XR Software module of the Cisco IOS XR System Security Configuration Guide</td>
</tr>
</tbody>
</table>

Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>draft-ietf-l2tpext-l2tp-base-03.txt</td>
<td>Layer Two Tunneling Protocol (Version 3)'L2TPv3'</td>
</tr>
</tbody>
</table>
MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1321</td>
<td><em>The MD5 Message Digest Algorithm</em></td>
</tr>
<tr>
<td>RFC 2104</td>
<td><em>HMAC-Keyed Hashing for Message Authentication</em></td>
</tr>
<tr>
<td>RFC 2661</td>
<td><em>Layer Two Tunneling Protocol “L2TP”</em></td>
</tr>
<tr>
<td>RFC 3931</td>
<td><em>Layer Two Tunneling Protocol Version 3 “L2TPv3”</em></td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.9.0</td>
<td>This feature was introduced.</td>
</tr>
</tbody>
</table>

Contents

- Prerequisites for Configuring MPLS VPNs over IP Tunnels, page VPC-170
- Restrictions for Configuring MPLS VPNs over IP Tunnels, page VPC-170
- Information About MPLS VPNs over IP Tunnels, page VPC-170
- How to Configure MPLS VPNs over IP Tunnels, page VPC-174
- Configuration Examples for MPLS VPNs over IP Tunnels, page VPC-189
- Additional References, page VPC-191
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.

- VPNv6 over L2TPv3 tunnel is currently not supported. Do not configure IPv6 or VPNv6 address family in the BGP configuration mode.

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, page VPC-171
- Advertising Tunnel Type and Tunnel Capabilities Between PE Routers—BGP, page VPC-171
- PE Routers and Address Space, page VPC-172
- 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. Packet Validation Mechanism, page VPC-172
- Quality of Service Using the Modular QoS CLI, page VPC-172
- BGP Multipath Load Sharing for MPLS VPNs over IP Tunnels, page VPC-172
- Inter-AS over IP Tunnels, page VPC-173
- Multiple Tunnel Source Address, page VPC-173
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.

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. **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 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.

**Note**
The Cisco CRS-1 router supports only the Inter-AS option A.

**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-174
- Tunnel EA, page VPC-174
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 FIB and 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-175 (required)
- Configuring a Route-Policy Definition, page VPC-177 (required)
- Configuring a Static Route, page VPC-177 (required)
- Configuring an IPv4 Loopback Interface, page VPC-179 (required)
- Configuring a CFI VRF Interface, page VPC-181 (required)
- Configuring the Core Network, page VPC-182 (required)
- Configuring Inter-AS over IP Tunnels, page VPC-183
- Verifying MPLS VPN over IP, page VPC-186 (optional)
- Configuring Source Pool Address for MPLS VPNs over IP Tunnels, page VPC-187 (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 ipv4 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
11. `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong>&lt;br&gt;Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>vrf vrf-name</strong>&lt;br&gt;Specifies a name assigned to a VRF.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config)# vrf vrf-name</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>address-family ipv4 unicast</strong>&lt;br&gt;Specifies an IPv4 address-family address.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>**import route-target [0-65535.0-65535:0-65535</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 500:99</td>
</tr>
</tbody>
</table>
Step 5: `export route-target [0-65535.0-65535:0-65535 | as-number:nn | ip-address:nn]`

Example:
RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 700:44

- **Purpose:** Configures a VPN routing and forwarding (VRF) export route-target extended community.

Step 6: `exit`

Example:
RP/0/RP0/CPU0:router(config-vrf-af)# exit

- **Purpose:** Exits interface configuration mode.

Step 7: `address-family ipv6 unicast`

Example:
RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast

- **Purpose:** Specifies an IPv4 address-family address.

Step 8: `import route-target [0-65535.0-65535:0-65535 | as-number:nn | ip-address:nn]`

Example:
RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 500:99

- **Purpose:** 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/RP0/CPU0:router(config-vrf-af)# export route-target 700:88

- **Purpose:** Configures a VPN routing and forwarding (VRF) export route-target extended community.

Step 10: `end`

or
`commit`

Example:
RP/0/RP0/CPU0:router(config-vrf-af)# end

or
RP/0/RP0/CPU0:router(config-vrf-af)# commit

- **Purpose:** 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. end policy

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> route-policy name pass</td>
<td>Defines and passes a route policy.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config)# route-policy ottawa_admin pass</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> end policy</td>
<td>End of route-policy definition.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-rpl)# end policy</td>
<td></td>
</tr>
</tbody>
</table>

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. end
   or
   commit
# How to Configure MPLS VPNs over IP Tunnels

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>
| **Example:**
RP/0/RP0/CPU0:router# configure    |                                                                         |
| **Step 2** router static           | Enters static route configuration subcommands.                          |
| **Example:**
RP/0/RP0/CPU0:router(config)# router static |                                                                         |
| **Step 3** maximum path ipv4 1-140000 | Enters the maximum number of static ipv4 paths that can be configured. |
| **Example:**
RP/0/RP0/CPU0:router (config-static)# maximum path ipv4 1-140000 |                                                                         |
| **Step 4** end or commit           | Saves configuration changes.                                            |
| **Example:**
RP/0/RP0/CPU0:router(config-static)# end or
RP/0/RP0/CPU0:router(config-static)# commit |                                                                         |

- 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> interface type interface-path-id</td>
<td>Enters interface configuration mode and enables a Loopback interface.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config)# interface Loopback0</td>
<td></td>
</tr>
</tbody>
</table>
### How to Configure MPLS VPNs over IP Tunnels

**Step 3**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipv4 address ipv4-address</code></td>
<td>Enters an IPv4 address and mask for the associated IP subnet. The network mask can be specified in either of two ways:</td>
</tr>
<tr>
<td>Example: <code>RP/0/RP0/CPU0:router(config-if)# ipv4 address 1.1.1.1 255.255.255.255</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
</tbody>
</table>

**Step 4**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>end</code> or <code>commit</code></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>Example: <code>RP/0/RP0/CPU0:router(config-if)# end</code> or <code>RP/0/RP0/CPU0:router(config-if)# commit</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When you issue the <code>end</code> command, the system prompts you to commit changes:</td>
</tr>
</tbody>
</table>
| | 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. `dot1q vlan vlan-id`
6. `end`
   or `commit`

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>configure</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>interface type interface-path-id</code></td>
<td>Enters interface configuration mode and enables a GigabitEthernet interface.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>interface GigabitEthernet0/0/0/1.1</code></td>
<td><code>RP/0/RP0/CPU0:router(config)# interface GigabitEthernet0/0/0/1.1</code></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>vrf vrf-name</code></td>
<td>Specifies a VRF name.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>vrf v1</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-if)# vrf v1</code></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>ipv4 address ipv4-address</code></td>
<td>Enters an IPv4 address and mask for the associated IP subnet. The network mask can be specified in either of two ways:</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>ipv4 address 100.1.10.2 255.255.255.0</code></td>
<td>- 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.</td>
</tr>
<tr>
<td></td>
<td>- 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.</td>
</tr>
</tbody>
</table>
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 over IP Tunnels

These tasks describe how to configure Inter-AS over IP tunnels:

- Configuring the ASBRs to Exchange VPN-IPv4 Addresses for IP Tunnels, page VPC-183
- Configuring the Backbone Carrier Core for IP Tunnels, page VPC-186

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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

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

<table>
<thead>
<tr>
<th>Step 2 router bgp autonomous-system-number</th>
<th>Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>address-family {ipv4 tunnel}</td>
<td>Configures IPv4 tunnel address family.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>address-family ipv4 tunnel</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>address-family {vpnv4 unicast}</td>
<td>Configures VPIVn4 address family.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>address-family vpng4 unicast</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>neighbor ip-address</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>neighbor 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>remote-as autonomous-system-number</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>remote-as 2002</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>address-family {vpnv4 unicast}</td>
<td>Configures VPIVn4 address family.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>address-family vpng4 unicast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>route-policy route-policy-name {in}</td>
<td>Applies a routing policy to updates that are received from a BGP neighbor.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>route-policy pass-all in</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>route-policy route-policy-name {out}</td>
<td>Applies a routing policy to updates that are sent from a BGP neighbor.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>route-policy pass-all out</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong>  neighbor ip-address</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong>  remote-as autonomous-system-number</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong>  update-source type interface-path-id</td>
<td>Allows BGP sessions to use the primary IP address from a particular interface as the local address.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong>  address-family {ipv4 tunnel}</td>
<td>Configures IPv4 tunnel address family.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong>  address-family {vpnv4 unicast}</td>
<td>Configures VPNv4 address family.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong>  end</td>
<td>Saves configuration changes.</td>
<td></td>
</tr>
<tr>
<td>or commit</td>
<td>- When you issue the <strong>end</strong> command, the system prompts you to commit changes:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Entering <strong>yes</strong> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Entering <strong>no</strong> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Entering <strong>cancel</strong> leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Use the <strong>commit</strong> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
<td></td>
</tr>
</tbody>
</table>
Configuring the Backbone Carrier Core for IP Tunnels

Configuring the backbone carrier core requires setting up connectivity and routing functions. To do so, you must complete the following high-level tasks:

- Verify IP connectivity.
- Configure IP tunnels in the core.
- Configure VRFs.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

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 rib ipv4 unicast opaques safi-tunnel bgp`
- `show tunnel-template tunnel-name`
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**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router# configure</code></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><code>tunnel-template name</code></td>
<td>Configures the tunnel template for source address.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config)#tunnel-template test</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem)#</code></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>mtu [mtu-value]</code></td>
<td>Configures the maximum transmission unit for the tunnel.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem) mtu 600</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem)#</code></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><code>ttl [ttl-value]</code></td>
<td>Configures the IP time to live (TTL).</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem)ttl 64</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem)</code></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><code>tos [tos-value]</code></td>
<td>Configures the tunnel header. By default, the TOS bits for the tunnel header are set to zero.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem)tos 7</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-tuntem)</code></td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 6</td>
<td><code>source loopback type interface-path-id</code></td>
<td>Configures the loopback interface.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-tuntem)source loopback0</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>source-pool A.B.C.D/prefix</code></td>
<td>Configures the source pool address.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-tuntem)# source-pool 10.10.10.0/28</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td><code>encapsulation l2tp</code></td>
<td>Configures the Layer 2 Tunnel Protocol encapsulation.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-tuntem)# encapsulation l2tp RP/0/RP0/CPU0:router(config-config-tunencap-l2tp)#</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td><code>end</code> or <code>commit</code></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-tuntem)# end</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-tuntem)# commit</td>
<td></td>
</tr>
</tbody>
</table>

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-189
- Configuring the Global VRF Definition: Example, page VPC-189
- Configuring a Route-Policy Definition: Example, page VPC-189
- Configuring a Static Route: Example, page VPC-190
- Configuring an IPv4 Loopback Interface: Example, page VPC-190
- Configuring a CFI VRF Interface: Example, page VPC-190
- Configuring Source Pool Address for MPLS VPNs over IP Tunnels: Example, page VPC-190

Configuring an L2TPv3 Tunnel: Example

The following example shows how to configure an L2TPv3 tunnel:

```plaintext
tunnel-template t1
  encapsulation l2tp

  source Loopback0

```

Configuring the Global VRF Definition: Example

The following example shows how to configure an L2TPv3 tunnel:

```plaintext
vrf v1
  address-family ipv4 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:

```plaintext
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>
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
  vrf v1
  ipv4 address 100.1.10.2 255.255.255.0
  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

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XR L2VPN command reference document</td>
<td>MPLS Virtual Private Network Commands on Cisco IOS XR Software</td>
</tr>
<tr>
<td>Layer 2 Tunnel Protocol Version 3</td>
<td>Layer 2 Tunnel Protocol Version 3 on Cisco IOS XR Software</td>
</tr>
<tr>
<td>Routing (BGP, EIGRP, OSPF, and RIP) commands: complete command syntax,</td>
<td>Cisco IOS XR Routing Command Reference</td>
</tr>
<tr>
<td>command modes, command history, defaults, usage guidelines, and examples</td>
<td></td>
</tr>
<tr>
<td>Routing (BGP, EIGRP, OSPF, and RIP) configuration</td>
<td>Cisco IOS XR Routing Configuration Guide</td>
</tr>
<tr>
<td>MPLS LDP configuration: configuration concepts, task, and examples</td>
<td>Implementing MPLS Label Distribution Protocol on Cisco IOS XR Software</td>
</tr>
<tr>
<td>MPLS Traffic Engineering Resource Reservation Protocol configuration:</td>
<td>Implementing RSVP for MPLS-TE and MPLS O-UNI on Cisco IOS XR Software</td>
</tr>
<tr>
<td>configuration concepts, task, and examples</td>
<td></td>
</tr>
<tr>
<td>Cisco CRS router getting started material</td>
<td>Cisco IOS XR Getting Started Guide</td>
</tr>
<tr>
<td>Information about user groups and task IDs</td>
<td>Configuring AAA Services on Cisco IOS XR Software module of the</td>
</tr>
<tr>
<td></td>
<td>Cisco IOS XR System Security Configuration Guide</td>
</tr>
</tbody>
</table>

### Standards

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support</td>
<td></td>
</tr>
<tr>
<td>for existing standards has not been modified by this feature.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>To locate and download MIBs using Cisco IOS XR software, use the Cisco</td>
</tr>
<tr>
<td></td>
<td>MIB Locator found at the following URL and choose a platform under the</td>
</tr>
<tr>
<td></td>
<td>Cisco Access Products menu:</td>
</tr>
</tbody>
</table>

Cisco IOS XR Virtual Private Network Configuration Guide for the Cisco CRS Router

OL-24669-01
RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 3931</td>
<td>Layer Two Tunneling Protocol - Version 3 (L2TPv3)</td>
</tr>
<tr>
<td>RFC 2547</td>
<td>BGP/MPLS VPNs</td>
</tr>
</tbody>
</table>

Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td><a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 3.3.0</td>
<td>This feature was introduced.</td>
</tr>
<tr>
<td>Release 3.4.0</td>
<td>Support was added for MPLS L3VPN Carrier Supporting Carrier (CSC) functionality, including conceptual information and configuration tasks.</td>
</tr>
<tr>
<td>Release 3.5.0</td>
<td>Support was added for 6VPE.</td>
</tr>
<tr>
<td></td>
<td>MPLS L3VPN Carrier Supporting Carrier (CSC) information was upgraded.</td>
</tr>
<tr>
<td>Release 3.6.0</td>
<td>Support was added for Inter-AS and CSC over IP Tunnels.</td>
</tr>
<tr>
<td>Release 3.7.0</td>
<td>Support was added for:</td>
</tr>
<tr>
<td></td>
<td>• I Pv6 VPN Provider Edge (6VPE).</td>
</tr>
<tr>
<td></td>
<td>• Inter-AS support for 6VPE.</td>
</tr>
<tr>
<td>Release 3.9.0</td>
<td>Support for Generic Routing Encapsulation (GRE) was added.</td>
</tr>
</tbody>
</table>
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-218 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**, page VPC-195
- **MPLS L3VPN Benefits**, page VPC-196
- **How MPLS L3VPN Works**, page VPC-197
- **MPLS L3VPN Major Components**, page VPC-199

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, 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 VPs.

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 \texttt{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 \texttt{rd auto} command assigns a Type 1 route distinguisher to the VRF using the following
format: \texttt{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 VPs. A given site can
be a member of multiple VPs. However, a site can associate with only one VRF. A customer-site VRF
contains all the routes available to the site from the VPs of which it is a member.

Inter-AS Support for L3VPN

This section contains the following topics:

- Inter-AS Restrictions, page VPC-200
- Inter-AS Support: Overview, page VPC-200
- Inter-AS and ASBRs, page VPC-200
- Transmitting Information Between Autonomous Systems, page VPC-201
- Exchanging VPN Routing Information, page VPC-202
- Packet Forwarding, page VPC-204
- Confederations, page VPC-207
- MPLS VPN Inter-AS BGP Label Distribution, page VPC-209
- Exchanging IPv4 Routes with MPLS labels, page VPC-209
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 does not get changed. 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 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

Figure 27 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.
**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 shows the same packet forwarding method, except the eBGP router (ASBR1) forwards the packet without reassigning a new label to it.

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.
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-258.

### MPLS VPN Inter-AS BGP Label Distribution

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

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.

Generic Routing Encapsulation Support for L3VPN

Generic Routing Encapsulation (GRE) is a tunneling protocol that can encapsulate many types of packets to enable data transmission using a tunnel. The GRE tunneling protocol enables:

- High assurance Internet Protocol encryptor (HAIPE) devices for encryption over the public Internet and nonsecure connections.
- Service providers (that do not run MPLS in their core network) to provide VPN services along with the security services.

GRE is used with IP to create a virtual point-to-point link to routers at remote points in a network. For detailed information about configuring GRE tunnel interfaces, see the `<module-name>` module of the *Cisco IOS XR Interfaces and Hardware Components Configuration Guide*.

**Note**

For a PE to PE (core) link, enable LDP (with implicit null) on the GRE interfaces for L3VPN.
GRE Restriction for L3VPN

The following restrictions are applicable to L3VPN forwarding over GRE:

- Carrier Supporting Carrier (CsC) or Inter-AS is not supported.
- GRE-based L3VPN does not interwork with MPLS or IP VPNs.
- GRE tunnel is supported only as a core link (PE-PE, PE-P, P-P, P-PE). A PE-CE (edge) link is not supported.
- VPPv6 forwarding using GRE tunnels is not supported.

VPNv4 Forwarding Using GRE Tunnels

This section describes the working of VPNv4 forwarding over GRE tunnels. The following description assumes that GRE is used only as a core link between the encapsulation and decapsulation provider edge (PE) routers that are connected to one or more customer edge (CE) routers.

Ingress of Encapsulation Router

On receiving prefixes from the CE routers, Border Gateway Protocol (BGP) assigns the VPN label to the prefixes that need to be exported. These VPN prefixes are then forwarded to the Forwarding Information Base (FIB) using the Route Information Base (RIB) or the label switched database (LSD). The FIB then populates the prefix in the appropriate VRF table. The FIB also populates the label in the global label table. Using BGP, the prefixes are then relayed to the remote PE router (decapsulation router).

Egress of Encapsulation Router

The forwarding behavior on egress of the encapsulation PE router is similar to the MPLS VPN label imposition. Regardless of whether the VPN label imposition is performed on the ingress or egress side, the GRE tunnel forwards a packet that has an associated label. This labeled packet is then encapsulated with a GRE header and forwarded based on the IP header.

Ingress of Decapsulation Router

The decapsulation PE router learns the VPN prefixes and label information from the remote encapsulation PE router using BGP. The next-hop information for the VPN prefix is the address of the GRE tunnel interface connecting the two PE routers. BGP downloads these prefixes to the RIB. The RIB downloads the routes to the FIB and the FIB installs the routes in the hardware.

Egress of Decapsulation Router

The egress forwarding behavior on the decapsulation PE router is similar to VPN disposition and forwarding, based on the protocol type of the inner payload.
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-213
- CSC Benefits, page VPC-213
- Configuration Options for the Backbone and Customer Carriers, page VPC-214

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 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-214).
  - An MPLS service provider with or without VPN services (see “Customer Carrier: MPLS Service Provider” section on page MPC-215).

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

![Network Diagram](image)

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

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-216).

This section includes the follow subsections:
- 6VPE Benefits, page VPC-216
- 6VPE Network Architecture, page VPC-216
- Dual Stack, page VPC-216
- 6VPE Operation, page VPC-217
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.

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.

**Note**

Multiple VRFs on the same physical or logical interface are not supported. Only one VRF, which is used for both IPv4 and IPv6 address families, is supported.

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-218
- Connecting MPLS VPN Customers, page VPC-221
- Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels, page VPC-242 (optional)
- Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses, page VPC-251 (optional)
- Configuring Carrier Supporting Carrier, page VPC-262 (optional)
- Verifying the MPLS Layer 3 VPN Configuration, page VPC-270
- Configuring L3VPN over GRE, page VPC-273

Configuring the Core Network

Configuring the core network includes the following tasks:

- Assessing the Needs of MPLS VPN Customers, page VPC-218
- Configuring Routing Protocols in the Core, page VPC-219
- Configuring MPLS in the Core, page VPC-219
- Determining if FIB Is Enabled in the Core, page VPC-219
- Configuring Multiprotocol BGP on the PE Routers and Route Reflectors, page VPC-220

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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters BGP configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><code>router bgp autonomous-system-number</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config)# router bgp 120</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.</td>
</tr>
</tbody>
</table>
| `address-family vpnv4 unicast`
   or
   `address-family vpnv6 unicast` | |
| **Example:** | RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast |
| **Step 4** | Creates a neighbor and assigns it a remote autonomous system number. |
| `neighbor ip-address remote-as autonomous-system-number` | |
| **Example:** | RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24 remote-as 2002 |
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-222
- Configuring VRF Interfaces on PE Routers for Each VPN Customer, page VPC-224
- Configuring BGP as the Routing Protocol Between the PE and CE Routers, page VPC-226 (optional)
- Configuring RIPv2 as the Routing Protocol Between the PE and CE Routers, page VPC-230 (optional)
- Configuring Static Routes Between the PE and CE Routers, page VPC-233 (optional)
- Configuring OSPF as the Routing Protocol Between the PE and CE Routers, page VPC-234 (optional)
- Configuring EIGRP as the Routing Protocol Between the PE and CE Routers, page VPC-237 (optional)
- Configuring EIGRP Redistribution in the MPLS VPN, page VPC-240 (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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> vrf vrf-name</td>
<td>Configures a VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# vrf vrf_1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family ipv4 unicast</td>
<td>Enters VRF address family configuration mode for the IPv4 address family.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> import route-policy policy-name</td>
<td>Specifies a route policy that can be imported into the local VPN.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-vrf-af)# import route-policy policy_A</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td>`import route-target [as-number:nn</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>export route-policy policy-name</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>`export route-target [as-number:nn</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><code>exit</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><code>exit</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><code>router bgp autonomous-system-number</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td><code>vrf vrf-name</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

Step 12

```
rd {as-number | ip-address | auto}
```

**Example:**

RP/0/RP0/CPU0:router(config-bgp-vrf)# rd auto

Step 13

```
end

or

commit
```

**Example:**

RP/0/RP0/CPU0:router(config-bgp-vrf)# end

or

RP/0/RP0/CPU0:router(config-bgp-vrf)# commit

Step 12

Automatically assigns a unique route distinguisher (RD) to vrf_1.

Step 13

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.
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> interface type interface-path-id</td>
<td>Enters interface configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/3/0/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> vrf vrf-name</td>
<td>Configures a VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-if)# vrf vrf_A</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> ipv4 address ipv4-address mask</td>
<td>Configures a primary IPv4 address for the specified interface.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end or commit</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-if)# end or RP/0/RP0/CPU0:router(config-if)# commit</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• When you issue the <strong>end</strong> command, the system prompts you to commit changes: Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
<td></td>
</tr>
<tr>
<td>• Entering <strong>yes</strong> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
<td></td>
</tr>
<tr>
<td>• Entering <strong>no</strong> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>• Entering <strong>cancel</strong> leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
<td></td>
</tr>
<tr>
<td>• Use the <strong>commit</strong> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><code>router bgp autonomous-system-number</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>Configures the local router with a router ID of 192.168.70.24.</td>
</tr>
<tr>
<td><code>bgp router-id (ip-address)</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for BGP routing.</td>
</tr>
<tr>
<td><code>vrf vrf-name</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>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.</td>
</tr>
<tr>
<td><code>label-allocation-mode per-ce</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Enters VRF address family configuration mode for the IPv4 address family.</td>
</tr>
<tr>
<td><code>address-family ipv4 unicast</code></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast</td>
<td></td>
</tr>
</tbody>
</table>
### Implementing MPLS Layer 3 VPNs

#### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 7    | `redistribute connected [metric metric-value]` | 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  
|      | `[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]` | 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. |
| 9    | `network {ip-address/prefix-length | ip-address mask} [route-policy route-policy-name]` | Configures the local router to originate and advertise the specified network. |
| 10   | `exit` | Exits VRF address family configuration mode and returns the router to VRF configuration mode for BGP routing. |

#### Example:

- **Step 7**
  - `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# redistribute connected`

- **Step 8**
  - `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/8 as-set`

- **Step 9**
  - `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# network 172.20.0.0/16`

- **Step 10**
  - `RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit`
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td><strong>neighbor</strong> <em>ip-address</em></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24</td>
</tr>
<tr>
<td>12</td>
<td><strong>remote-as</strong> <em>autonomous-system-number</em></td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2002</td>
</tr>
<tr>
<td>13</td>
<td><strong>password</strong> {<strong>clear</strong></td>
<td><strong>encrypted</strong>} <em>password</em></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# password clear pswd123</td>
</tr>
<tr>
<td>14</td>
<td><strong>ebgp-multihop</strong> [<em>ttl-value</em>]</td>
<td>Allows a BGP connection to neighbor 172.168.40.24.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop</td>
</tr>
<tr>
<td>15</td>
<td><strong>address-family</strong> ipv4 unicast</td>
<td>Enters VRF neighbor address family configuration mode for BGP routing.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast</td>
</tr>
<tr>
<td>16</td>
<td><strong>allowas-in</strong> [as-occurrence-number]</td>
<td>Replaces the neighbor autonomous system number (ASN) with the PE ASN in the AS path three times.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 3</td>
</tr>
<tr>
<td>17</td>
<td><strong>route-policy</strong> route-policy-name in</td>
<td>Applies the In-Ipv4 policy to inbound IPv4 unicast routes.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in</td>
</tr>
</tbody>
</table>
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

Step 18

**Command or Action**

route-policy route-policy-name out

**Purpose**

Applies the In-Ipv4 policy to outbound IPv4 unicast routes.

**Example:**

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)#
route-policy In-Ipv4 in
```

Step 19

**Command or Action**

end

or

commit

**Purpose**

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.
**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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router rip</code></td>
<td>Enters the Routing Information Protocol (RIP) configuration mode allowing you to configure the RIP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# router rip</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>vrf vrf-name</code></td>
<td>Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for RIP routing.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-rip)# vrf vrf_1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>interface type instance</code></td>
<td>Enters VRF interface configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-rip-vrf)# interface GigabitEthernet 0/3/0/0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> `site-of-origin {as-number:number</td>
<td>ip-address:number}`</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-rip-vrf-if)# site-of-origin 200:1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> <code>exit</code></td>
<td>Exits VRF interface configuration mode, and returns the router to VRF configuration mode for RIP routing.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-rip-vrf-if)# exit</td>
<td></td>
</tr>
</tbody>
</table>
### Step 7

**Command or Action**

```
redistribute bgp as-number [[external | internal | local] [route-policy name]]
or
redistribute connected [route-policy name]
or
redistribute eigrp as-number [route-policy name]
or
redistribute isis process-id [level-1 | level-1-2 | level-2] [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]
```

**Purpose**

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

**Example:**

```
RP/0/RP0/CPU0:router(config-rip-vrf)#
redistribute connected
```

### Step 8

**Command or Action**

```
end
or
commit
```

**Purpose**

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.

**Example:**

```
RP/0/RP0/CPU0:router(config-rip-vrf)# end
or
RP/0/RP0/CPU0:router(config-rip-vrf)# commit
```
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>router static</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config)# router static</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>vrf vrf-name</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-static)# vrf vrf_1</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>address-family ipv4 unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-static-vrf)# address-family ipv4 unicast</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

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

| **Step 2** router ospf process-name | Enters OSPF configuration mode allowing you to configure the OSPF routing process. |

**Example:**
```
RP/0/RP0/CPU0:router(config)# router ospf 109
```

| **Step 3** vrf vrf-name | Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing. |

**Example:**
```
RP/0/RP0/CPU0:router(config-ospf)# vrf vrf_1
```

| **Step 4** router-id {router-id | type interface-path-id} | Configures the router ID for the OSPF routing process. |

**Example:**
```
RP/0/RP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10
```
### How to Implement MPLS Layer 3 VPNs

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong></td>
<td>Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are:</td>
</tr>
</tbody>
</table>
| `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]` | • Border Gateway Protocol (BGP)  
• Connected  
• Enhanced Interior Gateway Routing Protocol (EIGRP)  
• OSPF  
• Static  
• Routing Information Protocol (RIP) |

**Example:**

RP/0/RP0/CPU0:router(config-ospf-vrf)#
`redistribute connected`

<table>
<thead>
<tr>
<th><strong>Step 6</strong></th>
<th>Configures the OSPF area as area 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>area area-id</code></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

RP/0/RP0/CPU0:router(config-ospf-vrf)# `area 0`
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 be 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router eigrp as-number</td>
<td>Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# router eigrp 24</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> vrf vrf-name</td>
<td>Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for EIGRP routing.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> address-family ipv4</td>
<td>Enters VRF address family configuration mode for the IPv4 address family.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-eigrp-vrf)# address family ipv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> router-id router-id</td>
<td>Configures the router ID for the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# router-id 172.20.0.0</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
</tr>
<tr>
<td>6</td>
<td><code>autonomous-system as-number</code></td>
</tr>
<tr>
<td>7</td>
<td><code>default-metric bandwidth delay reliability loading mtu</code></td>
</tr>
<tr>
<td>8</td>
<td>`redistribute {bgp</td>
</tr>
<tr>
<td>9</td>
<td><code>interface type interface-path-id</code></td>
</tr>
<tr>
<td>10</td>
<td>`site-of-origin {as-number:number</td>
</tr>
</tbody>
</table>
| 11   | `end` or `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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td>Step 2 router eigrp as-number</td>
<td>Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config)# router eigrp 24</td>
</tr>
<tr>
<td>Step 3 vrf vrf-name</td>
<td>Configures a VRF instance and enters VRF configuration mode for EIGRP routing.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1</td>
</tr>
</tbody>
</table>
### Command or Action

**Step 4**  
`address-family ipv4`  
**Example:**  
`RP/0/RP0/CPU0:router(config-eigrp-vrf)# address family ipv4`

**Step 5**  
`redistribute bgp [as-number] [route-policy policy-name]`  
**Example:**  
`RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 24 route-policy policy_A`

**Step 6**  
`end`  
or  
`commit`  
**Example:**  
`RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# end`  

or  
`RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# commit`

### Purpose

**Step 4**  
Enters VRF address family configuration mode for the IPv4 address family.

**Step 5**  
Causes Border Gateway Protocol (BGP) routes to be redistributed into EIGRP.

**Step 6**  
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-243
- Configuring the Route Reflectors to Exchange VPN-IPv4 Routes, page VPC-245
- Configuring the Route Reflector to Reflect Remote Routes in its AS, page VPC-248
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`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> <code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RP0/CPU0:router# configure</strong></td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router bgp autonomous-system-number</code></td>
<td>Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RP0/CPU0:router(config)# router bgp 120</strong></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>address-family ipv4 unicast</code></td>
<td>Enters global address family configuration mode for the IPv4 unicast address family.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</strong></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>allocate-label all</code></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td><strong>RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all</strong></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| **Step 5**
neighbor ip-address | Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer. |
| **Example:**
RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)# |
| **Step 6**
remote-as autonomous-system-number | Creates a neighbor and assigns it a remote autonomous system number. |
| **Example:**
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002 |
| **Step 7**
address-family ipv4 labeled-unicast | Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family. |
| **Example:**
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af) |
| **Step 8**
route-policy route-policy-name in | Applies a routing policy to updates that are received from a BGP neighbor. |
| **Example:**
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
route-policy pass-all in |
| • 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. |
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

### Configuring the Route Reflectors to Exchange VPN-IPv4 Routes

To configure the route reflectors to exchange VPN-IPv4 routes, perform the following steps:

**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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp autonomous-system-number</td>
<td>Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> neighbor ip-address</td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> remote-as autonomous-system-number</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> ebgp-multihop [ttl-value]</td>
<td>Enables multihop peerings with external BGP neighbors.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td>ebgp-multihop</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> update-source type interface-path-id</td>
<td>Allows BGP sessions to use the primary IP address from a particular interface as the local address.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td>update-source loopback0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> address-family vpnv4 unicast</td>
<td>Configures VPNv4 address family.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td>address-family vpnv4 unicast</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr#af)#</td>
<td></td>
</tr>
</tbody>
</table>
## Implementing MPLS Layer 3 VPNs

### How to Implement MPLS Layer 3 VPNs

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 8    | `route-policy route-policy-name 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 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. |
|      | **Example:**  
```bash
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
route-policy pass-all in
``` |
| 9    | `route-policy route-policy-name 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 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. |
|      | **Example:**  
```bash
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
rout-policy pass-all out
``` |
| 10   | `next-hop-unchanged` | Disables overwriting of the next hop before advertising to external Border Gateway Protocol (eBGP) peers. |
|      | **Example:**  
```bash
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
next-hop-unchanged
``` |
| 11   | `end` or `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. |
|      | **Example:**  
```bash
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end
```  
```bash
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit
``` |
## 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

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RP/0/RP0/CPU0:router# configure</strong></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><code>router bgp autonomous-system-number</code></td>
<td>Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RP/0/RP0/CPU0:router(config)# router bgp 120</strong></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>address-family ipv4 unicast</code></td>
<td>Enters global address family configuration mode for the IPv4 unicast address family.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>RP/0/RP0/CPU0:router(config-bgp-af)#</strong></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4</td>
<td>allocate-label all</td>
<td>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.</td>
</tr>
<tr>
<td>5</td>
<td>neighbor ip-address</td>
<td>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.</td>
</tr>
<tr>
<td>6</td>
<td>remote-as autonomous-system-number</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td>7</td>
<td>update-source type interface-path-id</td>
<td>Allows BGP sessions to use the primary IP address from a particular interface as the local address.</td>
</tr>
<tr>
<td>8</td>
<td>address-family ipv4 labeled-unicast</td>
<td>Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.</td>
</tr>
<tr>
<td>9</td>
<td>route-reflector-client</td>
<td>Configures the router as a BGP route reflector and neighbor 172.168.40.24 as its client.</td>
</tr>
<tr>
<td>10</td>
<td>neighbor ip-address</td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 10.25.40.2 as an VPNv4 iBGP peer.</td>
</tr>
<tr>
<td>11</td>
<td>remote-as autonomous-system-number</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
</tbody>
</table>
## Command or Action

**Step 12**  
**update-source type interface-path-id**

**Example:**
RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0

**Purpose:** 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/RP0/CPU0:router(config-bgp-nbr)#  
address-family ipv4 labeled-unicast  
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#

**Purpose:** Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

**Step 14**  
**route-reflector-client**

**Example:**
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#  
route-reflector-client

**Purpose:** Configures the neighbor as a route reflector client.

**Step 15**
**end**

or

**commit**

**Example:**
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end  
or
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit

**Purpose:** 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-251
- Configuring a Static Route to an ASBR Peer, page VPC-254
- Configuring EBGP Routing to Exchange VPN Routes Between Subautonomous Systems in a Confederation, page VPC-256
- Configuring MPLS Forwarding for ASBR Confederations, page VPC-258
- Configuring a Static Route to an ASBR Confederation Peer, page VPC-260

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. retain route-target { all | route-policy route-policy-name }
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 vpnv4 unicast
14. end
   or
   commit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router bgp autonomous-system-number</code></td>
<td>Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config)# router bgp 120</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>address-family vpnv4 unicast</code></td>
<td>Configures VPNv4 address family.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>address-family vpnv4 unicast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>`retain route-target (all</td>
<td>route-policy route-policy-name)`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>retain route-target route-policy policy1</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>neighbor ip-address</code></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>neighbor 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>remote-as autonomous-system-number</code></td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>remote-as 2002</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>address-family vpnv4 unicast</code></td>
<td>Configures VPNv4 address family.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>address-family vpnv4 unicast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>route-policy route-policy-name in</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in</td>
<td>Applies a routing policy to updates that are received from a BGP neighbor.&lt;ul&gt;&lt;li&gt;Use the <code>route-policy-name</code> argument to define the name of the route policy. The example shows that the route policy name is defined as pass-all.&lt;/li&gt;&lt;li&gt;Use the <code>in</code> keyword to define the policy for inbound routes.&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td><strong>route-policy route-policy-name out</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out</td>
<td>Applies a routing policy to updates that are sent from a BGP neighbor.&lt;ul&gt;&lt;li&gt;Use the <code>route-policy-name</code> argument to define the name of the route policy. The example shows that the route policy name is defined as pass-all.&lt;/li&gt;&lt;li&gt;Use the <code>out</code> keyword to define the policy for outbound routes.&lt;/li&gt;&lt;/ul&gt;</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td><strong>neighbor ip-address</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RP0/CPU0:router(config-bgp-nbr-af)# neighbor 10.40.25.2</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td><strong>remote-as autonomous-system-number</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td><strong>update-source type interface-path-id</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0</td>
<td>Allows BGP sessions to use the primary IP address from a particular interface as the local address.</td>
</tr>
</tbody>
</table>
## 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
6. `commit`

### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 13</td>
<td><code>address-family vpnv4 unicast</code></td>
<td>Configures VPNv4 address family.</td>
</tr>
</tbody>
</table>

**Example:**

RP/0/RP0/CPU0:router(config-bgp-nbr)#
`address-family vpnv4 unicast`
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#

Step 14

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| `end`
   or
| `commit` | Saves configuration changes. |

**Example:**

RP/0/RP0/CPU0:router(config-bgp-nbr-af)# `end`

or

RP/0/RP0/CPU0:router(config-bgp-nbr-af)# `commit`

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.
**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router static</td>
<td>Enters router static configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# router static RP/0/RP0/CPU0:router(config-static)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family ipv4 unicast</td>
<td>Enables an IPv4 address family.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast RP/0/RP0/CPU0:router(config-static-afi)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> A.B.C.D/length next-hop</td>
<td>Enters the address of the destination router (including IPv4 subnet mask).</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> end or commit</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-static-afi)# end or RP/0/RP0/CPU0:router(config-static-afi)# commit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When you issue the <strong>end</strong> command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td></td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
</tr>
<tr>
<td></td>
<td>• Entering <strong>yes</strong> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>• Entering <strong>no</strong> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• Entering <strong>cancel</strong> leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• Use the <strong>commit</strong> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router bgp autonomous-system-number</td>
<td>Enters BGP configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)#</td>
<td></td>
</tr>
<tr>
<td>router bgp 120</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 3</td>
<td><code>bgp confederation peers peer autonomous-system-number</code></td>
<td>Configures the peer autonomous system number that belongs to the confederation.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 8</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>bgp confederation identifier autonomous-system-number</code></td>
<td>Specifies the autonomous system number for the confederation ID.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>address-family vpnv4 unicast</code></td>
<td>Configures VPNv4 address family.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast RP/0/RP0/CPU0:router(config-bgp-af)#</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>neighbor ip-address</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 10.168.40.24 as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 10.168.40.24 RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>remote-as autonomous-system-number</code></td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td><code>address-family vpnv4 unicast</code></td>
<td>Configures VPNv4 address family.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td><code>route-policy route-policy-name in</code></td>
<td>Applies a routing policy to updates received from a BGP neighbor.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in</td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td><code>route-policy route-policy-name out</code></td>
<td>Applies a routing policy to updates advertised to a BGP neighbor.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy Out-Ipv4 out</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 11 next-hop-self</td>
<td>Disables next-hop calculation and let you insert your own address in the next-hop field of BGP updates.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-nbr-af)# next-hop-self</td>
<td></td>
</tr>
<tr>
<td>Step 12 end or commit</td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end or RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When you issue the end command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td></td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
</tr>
<tr>
<td></td>
<td>– Entering yes saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>– Entering no exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>– Entering cancel leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td>• Use the commit command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
</tbody>
</table>
## Detailed Steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure</code></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>2</td>
<td><code>router bgp as-number</code></td>
<td>Enters BGP configuration mode allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>3</td>
<td><code>mpls activate</code></td>
<td>Enters BGP MPLS activate configuration mode.</td>
</tr>
<tr>
<td>4</td>
<td><code>interface type interface-path-id</code></td>
<td>Enables MPLS on the interface.</td>
</tr>
</tbody>
</table>
| 5     | `end` or `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-268.

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router static</td>
<td>Enters router static configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router static</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-static)#</td>
<td></td>
</tr>
<tr>
<td>Step 3 address-family ipv4 unicast</td>
<td>Enables an IPv4 address family.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-static)#</td>
<td></td>
</tr>
<tr>
<td>address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-static-afi)#</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

**Step 4**  
**A.B.C.D/length**  
*next-hop*  

**Example:**  
RP/0/RP0/CPU0:router(config-static-afi)#  
10.10.10.32 10.9.9.9

**Purpose**  
Enters the address of the destination router (including IPv4 subnet mask).

**Step 5**  
*end*  
or  
*commit*  

**Example:**  
RP/0/RP0/CPU0:router(config-static-afi)# end  
or  
RP/0/RP0/CPU0:router(config-static-afi)# commit

**Purpose**  
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-262
- Configuring the Backbone Carrier Core, page VPC-263
- Configuring the CSC-PE and CSC-CE Routers, page VPC-263
- Configuring a Static Route to a Peer, page VPC-268

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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> Identify the type of customer carrier, ISP, or MPLS VPN service provider.</td>
<td>Sets up requirements for configuration of carrier supporting carrier network.</td>
</tr>
<tr>
<td><strong>Step 2</strong> Identify the CE routers.</td>
<td>Sets up requirements for configuration of CE to PE connections.</td>
</tr>
<tr>
<td><strong>Step 3</strong> Identify the customer carrier core router configuration.</td>
<td>Sets up requirements for configuration between core (P) routers and between P routers and edge routers (PE and CSC-CE routers).</td>
</tr>
<tr>
<td><strong>Step 4</strong> Identify the customer carrier edge (CSC-CE) routers.</td>
<td>Sets up requirements for configuration of CSC-CE to CSC-PE connections.</td>
</tr>
<tr>
<td><strong>Step 5</strong> Identify the backbone carrier router configuration.</td>
<td>Sets up requirements for configuration between CSC core routers and between CSC core routers and edge routers (CSC-CE and CSC-PE routers).</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>router bgp as-number</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config)# router bgp 2 RP/0/RP0/CPU0:router(config-bgp)#</td>
</tr>
<tr>
<td></td>
<td>Configures a BGP routing process and enters router configuration mode.</td>
</tr>
<tr>
<td></td>
<td>• Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>address-family vpnv4 unicast</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast RP/0/RP0/CPU0:router(config-bgp-af)#</td>
</tr>
<tr>
<td></td>
<td>Configures VPNv4 address family.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>neighbor A.B.C.D</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 10.10.10.0 RP/0/RP0/CPU0:router(config-bgp-nbr)#</td>
</tr>
<tr>
<td></td>
<td>Configures the IP address for the BGP neighbor.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>remote-as as-number</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 888</td>
</tr>
<tr>
<td></td>
<td>Configures the AS number for the BGP neighbor.</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>update-source type interface-path-id</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)# update-source loopback0</td>
</tr>
<tr>
<td></td>
<td>Allows BGP sessions to use the primary IP address from a particular interface as the local address.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 7</strong> address-family vpnv4 unicast</td>
<td>Configures VPNv4 unicast address family.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast RP/0/RP0/CPU0:router(config-bgp-nbr-af)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> vrf vrf-name</td>
<td>Configures a VRF instance.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-nbr-af)# vrf 9999 RP/0/RP0/CPU0:router(config-bgp-vrf)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> rd (as-number:nn</td>
<td>Configures a route distinguisher.</td>
</tr>
<tr>
<td></td>
<td>Note Use the <code>auto</code> keyword to automatically assign a unique route distinguisher.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Step 10</strong> address-family ipv4 unicast</td>
</tr>
<tr>
<td></td>
<td>Allocates labels for all local prefixes and prefixes received with labels.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf-af)# rd auto</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Step 11</strong> allocate-label all</td>
</tr>
<tr>
<td></td>
<td>Configures the IP address for the BGP neighbor.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf-af)# allocate-label all</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Step 13</strong> remote-as as-number</td>
</tr>
<tr>
<td></td>
<td>Enables the exchange of information with a neighboring BGP router.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 888</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Step 14</strong> address-family ipv4 labeled-unicast</td>
</tr>
<tr>
<td></td>
<td>Configures IPv4 labeled-unicast address family.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# address-family ipv4 labeled-unicast RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)#</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp as-number</td>
<td>Configures a BGP routing process and enters router configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# router bgp 1</td>
<td></td>
</tr>
<tr>
<td><strong>Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family ipv4 unicast</td>
<td>Configures IPv4 unicast address-family.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> redistribute ospf instance-number</td>
<td>Redistributes OSPF routes into BGP.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-router-af)# redistribute ospf 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> allocate-label route-policy route-policy-name</td>
<td>Allocates labels for those routes that match the route policy. These labeled routes are advertised to neighbors configured with address-family ipv4 labeled-unicast.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-router-af)# allocate-label route-policy internal-routes</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits the current configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-af)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> neighbor A.B.C.D</td>
<td>Configures the IP address for the BGP neighbor.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.0.0.1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> remote-as as-number</td>
<td>Enables the exchange of information with a neighboring BGP router.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1</td>
<td></td>
</tr>
</tbody>
</table>
## 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.

### Command or Action | Purpose
--- | ---
**Step 9** | address-family ipv4 labeled-unicast
Example:
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
| Configures IPv4 labeled-unicast address family. |

**Step 10** | route-policy route-policy-name in
Example:
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config-bgp)# end
or
RP/0/RP0/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.
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router static</td>
<td>Enters router static configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# router static</td>
<td></td>
</tr>
<tr>
<td>Step 3 address-family ipv4 unicast</td>
<td>Enables an IPv4 address family.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

To configure a static route on a CSC-PE, you must first configure the VRF using the `vrf` command before `address-family`. 
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> show running-config router bgp as-number vrf vrf-name</td>
<td>Displays the specified VPN routing and forwarding (VRF) content of the currently running configuration.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> show running-config routes</td>
<td>Displays the Open Shortest Path First (OSPF) routes table in the currently running configuration.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# show running-config routes</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> show ospf vrf vrf-name database</td>
<td>Displays lists of information related to the OSPF database for a specified VRF.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# show ospf vrf vrf_A database</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> show running-config router bgp as-number vrf vrf-name neighbor ip-address</td>
<td>Displays the Border Gateway Protocol (BGP) VRF neighbor content of the currently running configuration.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A neighbor 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> show bgp vrf vrf-name summary</td>
<td>Displays the status of the specified BGP VRF connections.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router# show bgp vrf vrf_A summary</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 6</strong> show bgp vrf vrf-name neighbors ip-address</td>
<td>Displays information about BGP VRF connections to the specified neighbors.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show bgp vrf vrf_A neighbors 172.168.40.24</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> show bgp vrf vrf-name</td>
<td>Displays information about a specified BGP VRF.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show bgp vrf vrf_A</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> show route vrf vrf-name ip-address</td>
<td>Displays the current routes in the Routing Information Base (RIB) for a specified VRF.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> show bgp vpn unicast summary</td>
<td>Displays the status of all BGP VPN unicast connections.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show bgp vpn unicast summary</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> show running-config router isis</td>
<td>Displays the Intermediate System-to-Intermediate System (IS-IS) content of the currently running configuration.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show running-config router isis</td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong> show running-config mpls</td>
<td>Displays the MPLS content of the currently running-configuration.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show running-config mpls</td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong> show isis adjacency</td>
<td>Displays IS-IS adjacency information.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show isis adjacency</td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong> show mpls ldp forwarding</td>
<td>Displays the Label Distribution Protocol (LDP) forwarding state installed in MPLS forwarding.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show mpls ldp forwarding</td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong> show bgp vpnv4 unicast or show bgp vpnv6 unicast</td>
<td>Displays entries in the BGP routing table for VPNv4 or VPNv6 unicast addresses.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show bgp vpnv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong> show bgp vrf vrf-name</td>
<td>Displays entries in the BGP routing table for VRF vrf_A.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# show bgp vrf vrf_A</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring L3VPN over GRE

Perform the following tasks to configure L3VPN over GRE:

- Creating a GRE Tunnel between Provider Edge Routers
- Configuring IGP between Provider Edge Routers
- Configuring LDP/GRE on the Provider Edge Routers
- Configuring L3VPN

### Creating a GRE Tunnel between Provider Edge Routers

Perform this task to configure a GRE tunnel between provider edge routers.

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 16</td>
<td>show bgp vrf vrf-name imported-routes</td>
<td>Displays BGP information for routes imported into specified VRF instances.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show bgp vrf vrf_A imported-routes</td>
<td></td>
</tr>
<tr>
<td>Step 17</td>
<td>show route vrf vrf-name ip-address</td>
<td>Displays the current specified VRF routes in the RIB.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0</td>
<td></td>
</tr>
<tr>
<td>Step 18</td>
<td>show cef vrf vrf-name ip-address</td>
<td>Displays the IPv4 Cisco Express Forwarding (CEF) table for a specified VRF.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1</td>
<td></td>
</tr>
<tr>
<td>Step 19</td>
<td>show cef vrf vrf-name ip-address location node-id</td>
<td>Displays the IPv4 CEF table for a specified VRF and location.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1 location 0/1/cpu0</td>
<td></td>
</tr>
<tr>
<td>Step 20</td>
<td>show bgp vrf vrf-name ip-address</td>
<td>Displays entries in the BGP routing table for VRF vrf_A.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show bgp vrf vrf_A 10.0.0.0</td>
<td></td>
</tr>
<tr>
<td>Step 21</td>
<td>show ospf vrf vrf-name database</td>
<td>Displays lists of information related to the OSPF database for a specified VRF.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# show ospf vrf vrf_A database</td>
<td></td>
</tr>
</tbody>
</table>
SUMMARY STEPS

1. configure
2. interface tunnel-ip number
3. ipv4 address ipv4-address subnet-mask
4. ipv6 address ipv6-prefix/prefix-length
5. tunnel mode gre ipv4
6. tunnel source type number
7. tunnel destination ip-address
8. end
   or
   commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> interface tunnel-ip number</td>
<td>Enters tunnel interface configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# interface tunnel-ip 4000</td>
<td>number is the number associated with the tunnel interface.</td>
</tr>
<tr>
<td><strong>Step 3</strong> ipv4 address ipv4-address subnet-mask</td>
<td>Specifies the IPv4 address and subnet mask for the interface.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0</td>
<td>ipv4-address specifies the IP address of the interface.</td>
</tr>
<tr>
<td></td>
<td>subnet-mask specifies the subnet mask of the interface.</td>
</tr>
<tr>
<td><strong>Step 4</strong> ipv6 address ipv6-prefix/prefix-length</td>
<td>Specifies an IPv6 network assigned to the interface.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# ipv6 address 100:1::1:1::1/64</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> tunnel mode gre ipv4</td>
<td>Sets the encapsulation mode of the tunnel interface to GRE.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# tunnel mode gre ipv4</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> tunnel source type path-id</td>
<td>Specifies the source of the tunnel interface.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring IGP between Provider Edge Routers

Perform this task to configure IGP between provider edge routers.

**SUMMARY STEPS**

1. `configure`
2. `router ospf process-name`
3. `nsr`
4. `router-id {router-id}`
5. `mpls ldp sync`
6. `dead-interval seconds`
7. `hello-interval seconds`
8. `area area-id`
9. `interface tunnel-ip number`
10. `end`
    or
11. `commit`

**Command or Action**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 7</strong></td>
<td>tunnel destination <em>ip-address</em></td>
</tr>
<tr>
<td>Defines the tunnel destination.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>end</td>
</tr>
<tr>
<td>Saves configuration changes.</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# tunnel destination 145.12.5.2
```

---

*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.*
## Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure</strong></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RP0/CPU0:router# configure</td>
</tr>
<tr>
<td></td>
<td>Enters global configuration mode.</td>
</tr>
</tbody>
</table>

| **Step 2** | **router ospf process-name**  |
| Example: | RP/0/RP0/CPU0:router(config)# router ospf 1 |
| | Enables OSPF routing for the specified routing process and places the router in router configuration mode. |

| **Step 3** | **nsr**  |
| Example: | RP/0/RP0/CPU0:router(config-ospf)# nsr |
| | Activates BGP NSR. |

| **Step 4** | **router-id (router-id)**  |
| Example: | RP/0/RP0/CPU0:router(config-ospf)# router-id 1.1.1.1 |
| | Configures a router ID for the OSPF process. |
| Note | We recommend using a stable IP address as the router ID. |

| **Step 5** | **mpls ldp sync**  |
| Example: | RP/0/RP0/CPU0:router(config-ospf)# mpls ldp sync |
| | Enables MPLS LDP synchronization. |

| **Step 6** | **dead-interval seconds**  |
| Example: | RP/0/RP0/CPU0:router(config-ospf)# dead-interval 60 |
| | Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down. |

| **Step 7** | **hello-interval seconds**  |
| Example: | RP/0/RP0/CPU0:router(config-ospf)# hello-interval 15 |
| | Specifies the interval between hello packets that OSPF sends on the interface. |

| **Step 8** | **area area-id**  |
| Example: | RP/0/RP0/CPU0:router(config-ospf)# area 0 |
| | Enters area configuration mode and configures an area for the OSPF process. |
Configuring LDP/GRE on the Provider Edge Routers

Perform this task to configure LDP/GRE on the provider edge routers.

SUMMARY STEPS

1. configure
2. mpls ldp
3. router-id [router-id]
4. discovery hello holdtime seconds
5. discovery hello interval seconds
6. nsr
7. graceful-restart
8. graceful-restart reconnect-timeout seconds
9. graceful-restart forwarding-state-holdtime seconds
10. holdtime seconds
11. neighbor ip-address
12. interface tunnel-ip number

Example:
RP/0/RP0/CPU0:router(config-ospf)# interface tunnel-ip 4

Step 9
interface tunnel-ip number

Purpose
Enters tunnel interface configuration mode.
- number is the number associated with the tunnel interface.

Example:
RP/0/RP0/CPU0:router(config-ospf)# interface tunnel-ip 4

Step 10
end or commit

Purpose
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.
### Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> mpls ldp</td>
<td>Enables MPLS LDP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config)# mpls ldp</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> router-id {router-id}</td>
<td>Configures a router ID for the OSPF process.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ldp)# router-id 1.1.1.1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> discovery hello holdtime seconds</td>
<td>Defines the period of time a discovered LDP neighbor is remembered without receipt of an LDP Hello message from the neighbor.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ldp)# discovery hello holdtime 40</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> discovery hello interval seconds</td>
<td>Defines the period of time between the sending of consecutive Hello messages.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ldp)# discovery hello holdtime 20</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> nsr</td>
<td>Activates BGP NSR.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ldp)# nsr</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> graceful-restart</td>
<td>Enables graceful restart on the router.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ldp)# graceful-restart</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> graceful-restart reconnect-timeout seconds</td>
<td>Defines the time for which the neighbor should wait for a reconnection if the LDP session is lost.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RP0/CPU0:router(config-ldp)# graceful-restart reconnect-timeout 180</td>
<td></td>
</tr>
</tbody>
</table>
## Configuring L3VPN

Perform this task to configure L3VPN.

### SUMMARY STEPS

1. `configure`
2. `vrf vrf-name`

---

### Command or Action | Purpose
---|---
**Step 9**

```
graceful-restart forwarding-state-holdtime
seconds
```

Defines the time that the neighbor should retain the MPLS forwarding state during a recovery.

**Example:**

```
RP/0/RP0/CPU0:router(config-ldp)# graceful-restart forwarding-state-holdtime 300
```

**Step 10**

```
holdtime seconds
```

Configures the hold time for an interface.

**Example:**

```
RP/0/RP0/CPU0:router(config-ldp)# holdtime 90
```

**Step 11**

```
neighbor ip-address
```

Defines a neighboring router.

**Example:**

```
RP/0/RP0/CPU0:router(config-ldp)# neighbor 10.1.1.0
```

**Step 12**

```
interface tunnel-ip number
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

**Example:**

```
RP/0/RP0/CPU0:router(config-ldp)# interface tunnel-ip 4
```

**Step 13**

```
end
```

or

```
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.

```
RP/0/RP0/CPU0:router(config-ldp)# end
```

or

```
RP/0/RP0/CPU0:router(config-ldp)# commit
```
3. address-family { ipv4 | ipv6 } unicast
4. import route-target [as-number:nn | ip-address:nn]
5. export route-target [as-number:nn | ip-address:nn]
6. interface type interface-path-id
7. vrf vrf-name
8. ipv4 address ipv4-address subnet-mask
9. dot1q vlan vlan-id
10. router bgp process-name
11. nsr
12. bgp router-id ip-address
13. address-family {vpn4 | vpnv6} unicast
14. neighbor ip-address
15. remote-as as-number
16. update-source type interface-path-id
17. address-family {vpn4 | vpnv6} unicast
18. route-policy policy-name in
19. route-policy policy-name out
20. vrf vrf-name
21. rd {as-number:nn | ip-address:nn | auto}
22. address-family {ipv4 | ipv6} unicast
23. redistribute connected [metric metric-value] [route-policy route-policy-name]
24. redistribute static [metric metric-value] [route-policy route-policy-name]
25. neighbor ip-address
26. remote-as as-number
27. ebgp-multihop ttl-value
28. address-family {ipv4 | ipv6} unicast
29. route-policy policy-name in
30. route-policy policy-name out
31. end
   or
   commit
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>vrf vrf-name</td>
<td>Configures a VRF instance.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config)# vrf vpn1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config-vrf)# address-family { ipv4</td>
<td>ipv6 } unicast</td>
</tr>
<tr>
<td>4</td>
<td>import route-target {as-number:nn</td>
<td>ip-address:nn}</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config-vrf)# import route-target 2:1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>export route-target {as-number:nn</td>
<td>ip-address:nn}</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config-vrf)# export route-target 1:1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>interface type interface-path-id</td>
<td>Enters interface configuration mode and configures an interface.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config)#interface TenGigE0/2/0/0.1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>vrf vrf-name</td>
<td>Configures a VRF instance.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config-if)# vrf vpn1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>ipv4 address ipv4-address subnet-mask</td>
<td>Specifies the IPv4 address and subnet mask for the interface.</td>
</tr>
<tr>
<td></td>
<td>• ipv4-address specifies the IP address of the interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• subnet-mask specifies the subnet mask of the interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config-if)# ipv4 address 150.1.1.1 255.255.255.0</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>dot1q native vlan vlan-id</td>
<td>Assigns the native VLAN ID of a physical interface trunking 802.1Q VLAN traffic.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RP0/CPU0:router(config-if)# dot1q native vlan 1</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 10</strong> <code>router bgp as-number</code></td>
<td>Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config)# router bgp 1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong> <code>nsr</code></td>
<td>Activates BGP NSR.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp)# nsr</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong> <code>bgp router-id ip-address</code></td>
<td>Configures the local router with a specified router ID.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong> `address-family {vpn4</td>
<td>vpnv6} unicast`</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv4 unicast</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong> <code>neighbor ip-address</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp)# neighbor 4.4.4.4</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong> <code>remote-as as-number</code></td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 16</strong> <code>update-source type interface-path-id</code></td>
<td>Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp-nbr)#update-source Loopback0</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 17</strong> `address-family {vpn4</td>
<td>vpnv6} unicast`</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 18</strong> <code>route-policy route-policy-name in</code></td>
<td>Defines a route policy and enters route policy configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy pass-all in</code></td>
<td></td>
</tr>
</tbody>
</table>
### Implementing MPLS Layer 3 VPNs

**How to Implement MPLS Layer 3 VPNs**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>route-policy route-policy-name out</td>
<td>Defines a route policy and enters route policy configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy pass-all out</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>vrf vrf-name</td>
<td>Configures a VRF instance.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config)# vrf vpn1</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>rd (as-number:nn</td>
<td>Configures the route distinguisher.</td>
</tr>
<tr>
<td></td>
<td>ip-address:nn</td>
<td></td>
</tr>
<tr>
<td></td>
<td>auto)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-vrf)#rd 1:1</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>address-family { ipv4</td>
<td>Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.</td>
</tr>
<tr>
<td></td>
<td>ipv6 } unicast</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>redistribute connected [metric metric-value] [route-policy route-policy-name]</td>
<td>Causes routes from the specified instance to be redistributed into BGP.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-vrf-af)# redistribute connected</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>redistribute static [metric metric-value] [route-policy route-policy-name]</td>
<td>Causes routes from the specified instance to be redistributed into BGP.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-vrf-af)# redistribute static</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>neighbor ip-address</td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp)# neighbor 150.1.1.2</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>remote-as as-number</td>
<td>Creates a neighbor and assigns a remote autonomous system number to it.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 7501</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>ebg-multihop ttl-value</td>
<td>Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp-nbr)#ebgp-multihop 10</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 28</td>
<td>`address-family { ipv4</td>
<td>ipv6 } unicast`</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</code></td>
</tr>
<tr>
<td>Step 29</td>
<td><code>route-policy route-policy-name in</code></td>
<td>Defines a route policy and enters route policy configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy BGP_pass_all in</code></td>
</tr>
<tr>
<td>Step 30</td>
<td><code>route-policy route-policy-name out</code></td>
<td>Defines a route policy and enters route policy configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy BGP_pass_all out</code></td>
</tr>
<tr>
<td>Step 31</td>
<td><code>end</code> or <code>commit</code></td>
<td>Saves configuration changes.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td><code>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# end</code> or <code>RP/0/RP0/CPU0:router(config-bgp-nbr-af)# commit</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- When you issue the <code>end</code> command, the system prompts you to commit changes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Entering <code>yes</code> saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Entering <code>no</code> exits the configuration session and returns the router to EXEC mode without committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Entering <code>cancel</code> leaves the router in the current configuration session without exiting or committing the configuration changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Use the <code>commit</code> command to save the configuration changes to the running configuration file and remain within the configuration session.</td>
</tr>
</tbody>
</table>

## Configuring 6VPE Support

The following tasks are required to configure 6VPE support:

- Configuring an IPv6 Address Family Under VRF, page VPC-285
- Configuring BGP Route Distinguisher and Core-facing Sessions, page VPC-286
- Configuring a PE-CE Protocol, page VPC-288
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> vrf vrf-name</td>
<td>Configures a VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family ipv6 unicast</td>
<td>Enters VRF address family configuration mode for the IPv6 address family.</td>
</tr>
<tr>
<td><strong>Step 4</strong> import route-target [as-number:nn</td>
<td>ip-address:nn]</td>
</tr>
</tbody>
</table>

**Example:**

- **Step 1**
  
  ```
  RP/0/RP0/CPU0:router# configure
  ```

- **Step 2**
  
  ```
  RP/0/RP0/CPU0:router(config)# vrf vrf_1
  ```

- **Step 3**
  
  ```
  RP/0/RP0/CPU0:router(config-vrf)# address-family ipv4 unicast
  ```

- **Step 4**
  
  ```
  RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 120:1
  ```
Configuring 6VPE Support

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 5**
`export route-target [as-number:nn | ip-address:nn]` | 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. |
| **Example:**
RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 120:2 | |
| **Step 6**
`end` or `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. |
| **Example:**
RP/0/RP0/CPU0:router(config-vrf-af)# end or RP/0/RP0/CPU0:router(config-vrf-af)# commit | |

---

**Configuring BGP Route Distinguisher and Core-facing Sessions**

Perform this task to configure VRF route distinguisher values and core-facing neighbors under BGP.

### 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router bgp as-number</td>
<td>Enters router BGP configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config)# router bgp 100 RP/0/RP0/CPU0:router(config-bgp)#</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> address-family vpnv6 unicast</td>
<td>Enters address family configuration mode for the VPNv6 address family.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv6 unicast RP/0/RP0/CPU0:router(config-bgp-af)</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> vrf vrf-name</td>
<td>Configures a VPN VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp)# vrf red</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> rd ( (as-number:nn \mid ip-address:nn \mid auto) )</td>
<td>Configures a route distinguisher.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf)# vrf red</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> address-family ipv6 unicast</td>
<td>Enters IPv6 address family configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> exit</td>
<td>Exits the current configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> neighbor ip-address remote-as as-number</td>
<td>Creates a neighbor and assigns it a remote autonomous system number.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24 remote-as 2002f</td>
<td></td>
</tr>
</tbody>
</table>
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 vvpn6 unicast
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router# configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>router bgp as-number</td>
<td>Enters router BGP configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config)# router bgp 120 RP/0/RP0/CPU0:router(config-bgp)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>vrf vrf-name</td>
<td>Configures a VPN VRF instance and enters VRF configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:router(config-bgp)# vrf red RP/0/RP0/CPU0:router(config-bgp-vrf)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>address-family ipv6 unicast</td>
<td>Enters IPv6 address family configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
|      | RP/0/RP0/CPU0:router(config-bgp-vrf)
address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-bgp-vrf-af) |         |
| 5    | exit              | Exits the current configuration mode. |
|      | **Example:**      |         |
|      | RP/0/RP0/CPU0:router(config-bgp-vrf-af)# exit |         |
| 6    | exit              | Exits the current configuration mode. |
|      | **Example:**      |         |
|      | RP/0/RP0/CPU0:router(config-bgp-vrf)# exit |         |
| 7    | neighbor ip-address | Creates a neighbor and assigns it a remote autonomous system number of 2002. |
|      | **Example:**      |         |
|      | RP/0/RP0/CPU0:router(config-bgp)# neighbor 10.10.10.10 |         |
| 8    | remote-as as-number | Creates a BGP neighbor and begin the exchange of routing information. |
|      | **Example:**      |         |
|      | RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1000 |         |
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-291
- Configuring the Routing Information Protocol on the PE Router: Example, page VPC-292
- Configuring the PE Router Using EIGRP: Example, page VPC-292
- Configuration Examples for MPLS VPN CSC, page VPC-292
- Configuration Examples for 6VPE, page VPC-296
Configuring an MPLS VPN Using BGP: Example

The following example shows the configuration for an MPLS VPN using BGP on “vrf vpn1”:

```conf
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
  retain route-target route-policy policy1
  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 loopback0
  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:

```conf
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:

```conf
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-293
- Configuring the Links Between CSC-PE and CSC-CE Routers: Examples, page VPC-293
- Configuring a Static Route to a Peer: Example, page VPC-294
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-293
- Configuring the Links Between CSC-PE and CSC-CE Routers: Examples, page VPC-293

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-293
- Configuring a CSC-CE: Example, page VPC-293

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
```
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:

```conf
config
router static
  address-family ipv4 unicast
    10.0.0.2/32 40.1.1.1
end
```

Configuring L3VPN over GRE: Example

The following example shows how to configure L3VPN over GRE:

Sample configuration to create a GRE tunnel between PE1 and PE2:

```conf
RP/0/RP0/CPU0:PE1#sh run int tunnel-ip 1
interface tunnel-ip1
  ipv4 address 100.1.1.1 255.255.255.0
  ipv6 address 100::1::1::1/64
  tunnel mode gre ipv4
  tunnel source TenGigE0/2/0/1
  tunnel destination 145.12.5.2

RP/0/RP0/CPU0:PE2#sh run int tunnel-ip 1
interface tunnel-ip1
  ipv4 address 100.1.1.2 255.255.255.0
  ipv6 address 100::1::1::2/64
  tunnel mode gre ipv4
  tunnel source TenGigE0/1/0/2
  tunnel destination 145.12.1.1
```

Configure IGP between PE1 and PE2:

Sample configuration for PE1 is given below. PE2 will also have a similar configuration.

```conf
RP/0/RP0/CPU0:PE1#sh run router ospf 1
router ospf 1
  nsr
  router-id 1.1.1.1 <=== Loopback0
  mpls ldp sync
  mtu-ignore enable
  dead-interval 60
  hello-interval 15
  area 0
  interface TenGigE0/2/0/1
  !
RP/0/RP0/CPU0:PE1#sh run router ospf 0
router ospf 0
  nsr
  router-id 1.1.1.1
  mpls ldp sync
```
dead-interval 60
hello-interval 15
area 0
    interface Loopback0
    !
    interface tunnel-ip1
    !

* Check for OSPF neighbors

```
RP/0/RP0/CPU0:PE1#sh ospf neighbor
Neighbors for OSPF 0
Neighbor ID     Pri   State           Dead Time   Address         Interface
4.4.4.4         1     FULL/  -        00:00:47    100.1.1.2       tunnel-ip1  <=
Neighbor PE2
    Neighbor is up for 00:13:40
Neighbors for OSPF 1
Neighbor ID     Pri   State           Dead Time   Address         Interface
2.2.2.2         1     FULL/DR         00:00:50    145.12.1.2      TenGigE0/2/0/1  <=
    Neighbor P1
    Neighbor is up for 00:13:43
```

Configure LDP/GRE on PE1 and PE2:

```
RP/0/RP0/CPU0:PE1#sh run mpls ldp
mpls ldp
    router-id 1.1.1.1 <= Loopback0
    discovery hello holdtime 45
discovery hello interval 15
nsr
    graceful-restart
    graceful-restart reconnect-timeout 180
    graceful-restart forwarding-state-holdtime 300
    holdtime 90
    log
    neighbor
    !
    interface tunnel-ip1
    !

*Check for mpls forwarding

```
RP/0/RP0/CPU0:PE1#sh run mpls forwarding prefix 4.4.4.4/32
Local Outgoing Prefix Outgoing Next Hop Bytes
Label Label or ID Interface Switched
---- ------- -------------- -------- ----------- ----------
16003 Pop 4.4.4.4/32 ti1 100.4.1.2 0

Configure L3VPN

```
RP/0/RP0/CPU0:PE1#sh run vrf vpn1
vrf vpn1
    address-family ipv4 unicast
    import route-target
    2:1
    !
    export route-target
    1:1
    !
RP/0/RP0/CPU0:PE1#sh run int tenGigE 0/2/0/0.1
interface TenGigE0/2/0/0.1
 vrf vpn1
 ipv4 address 150.1.1.1 255.255.255.0
dot1q vlan 1
!

RP/0/RP0/CPU0:PE1#sh run router bgp
router bgp 1
 nsr
 bgp router-id 1.1.1.1 <===Loopback0
 address-family vpnv4 unicast
 !
 neighbor 4.4.4.4 <==ibGP session with PE2
remote-as 1
 update-source Loopback0
 address-family vpnv4 unicast
 route-policy pass-all in
 route-policy pass-all out
 !
!
 vrf vpn1
rd 1:1
 address-family ipv4 unicast
 redistribute connected
 redistribute static
!
 neighbor 150.1.1.2 <== VRF neighbor
remote-as 7501
ebgp-multihop 10
 address-family ipv4 unicast
 route-policy BGP_pass_all in
 route-policy BGP_pass_all out
!
*
Check vrf ping to the 150.1.1.2.

RP/0/RP0/CPU0:PE1#ping vrf vpn1 150.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms

* Send traffic to vrf routes advertised and verify that mpls counters increase in tunnel
 interface accounting

RP/0/RP0/CPU0:PE1#sh int tunnel-ip1 accounting
tunnel-ip1
 Protocol        Pkts In  Chars In  Pkts Out  Chars Out
IPV4_MULTICAST  3        276      3         276
MPLS           697747    48842290  0          0

Configuration Examples for 6VPE

Configuration examples for the MPLS VPN CSC include:
- Configuring an IPv6 Address Family Under VRF: Example, page VPC-297
- Configuring BGP for the Address Family VPNv6: Example, page VPC-297
- Configuring a PE-CE Protocol: Example, page VPC-297
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, E - 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, E - 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

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XR L2VPN command reference document</td>
<td><em>MPLS Virtual Private Network Commands on Cisco IOS XR</em>&lt;br&gt;<em>Software module in Cisco IOS XR MPLS Configuration Guide</em></td>
</tr>
<tr>
<td>Routing (BGP, EIGRP, OSPF, and RIP) commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples</td>
<td><em>Cisco IOS XR Routing Command Reference</em></td>
</tr>
<tr>
<td>Routing (BGP, EIGRP, OSPF, and RIP) configuration</td>
<td><em>Cisco IOS XR Routing Configuration Guide</em></td>
</tr>
</tbody>
</table>
Implementing MPLS Layer 3 VPNs

Related Topic | Document Title
---|---
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

<table>
<thead>
<tr>
<th>Standards</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
<td>---</td>
</tr>
</tbody>
</table>

MIBs

<table>
<thead>
<tr>
<th>MIBs</th>
<th>MIBs Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a></td>
</tr>
</tbody>
</table>

RFCs

<table>
<thead>
<tr>
<th>RFCs</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 1700</td>
<td>Assigned Numbers</td>
</tr>
<tr>
<td>RFC 1918</td>
<td>Address Allocation for Private Internets</td>
</tr>
<tr>
<td>RFC 1966</td>
<td>BGP Route Reflectors: An Alternative to Full Mesh iBGP</td>
</tr>
<tr>
<td>RFC 2283</td>
<td>Multiprotocol Extensions for BGP-4</td>
</tr>
<tr>
<td>RFC 2547</td>
<td>BGP/MPLS VPNs</td>
</tr>
<tr>
<td>RFC 2842</td>
<td>Capabilities Advertisement with BGP-4</td>
</tr>
<tr>
<td>RFC 2858</td>
<td>Multiprotocol Extensions for BGP-4</td>
</tr>
<tr>
<td>RFC 3107</td>
<td>Carrying Label Information in BGP-4</td>
</tr>
</tbody>
</table>
Technical Assistance

<table>
<thead>
<tr>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>able technical content, including links to products, technologies,</td>
<td></td>
</tr>
<tr>
<td>solutions, technical tips, and tools. Registered Cisco.com users can</td>
<td></td>
</tr>
<tr>
<td>log in from this page to access even more content.</td>
<td></td>
</tr>
</tbody>
</table>
INDEX

Numerics

6PE
- BGP multipath VPC-130
  - conditions for use VPC-128
- configuring VPC-131
- deploying VPC-129
- overview VPC-128
- prerequisites VPC-128
- service provider considerations VPC-128
- supported protocols VPC-132

6VPE
- configuration tasks VPC-282
- configuring PE-CE VPC-286
- configuring VRFs VPC-284

A
- address space, PE routers VPC-172
- advertising tunnels, VPN services VPC-171
- aging, MAC address
  - configuring VPC-95
  - defining VPC-54

B
- BGP
  - confederations VPC-201
  - BGP (border gateway protocol)
    - distributing routes VPC-211
    - messages and MPLS labels VPC-211
    - routing information VPC-210
    - SSA function VPC-171
  - BGP4 configuration VPC-128
  - BGP multipath
    - 6PE VPC-130
    - load sharing VPC-172
  - bridge domain
    - associating members VPC-70
    - configuring parameters VPC-72
    - creating VPC-58
    - disabling VPC-73
    - overview VPC-53
    - split horizon VPC-53

how to define VPC-54

AToM (any transport over MPLS)
- IP Interworking VPC-25
- static pseudowire VPC-83
- using static labels VPC-83

attachment circuits
- defining VPC-48

automatic route renderer, MPLS Layer 3

VPN VPC-199

autonomous system VPC-200
CEF (Cisco Express Forwarding)
- L2TPv3 prerequisite  VPC-139
CEF (Cisco Express Forwarding)
- L2TPv3 prerequisite  VPC-139
CE-PE eBGP, configuring route-policy definition  VPC-177
CFI VRF interface, configuring  VPC-181
configuration examples
- MPLS L2VPN  VPC-37
control-channels
- configuring authentication parameters  VPC-150
- configuring L2TP maintenance parameters  VPC-159
- configuring method of authentication  VPC-151
- configuring timing parameters  VPC-149
- L2TP maintenance parameters, how to configure  VPC-159
- message authentication  VPC-150
- parameters, how to configure  VPC-148
- timing parameters, how to configure  VPC-149
core network, how to configure  VPC-182
CSC (Carrier Supporting Carrier)
- configuration examples  VPC-266
- configuration options for backbone and customer carriers  VPC-214
- configuring a CSC-PE link  VPC-261
- configuring a static route to a peer  VPC-266
- customer carrier network options  VPC-214
- identifying topology  VPC-260
CSC-CE link, how to configure  VPC-264
CSC-PE link, how to configure  VPC-261
customer edge router
- 6PE  VPC-129
- MPLS Layer 3 VPN  VPC-196
customer edge router (CE)
- MPLS Layer 3 VPN  VPC-196
digest secret graceful switchover, L2TPv3  VPC-154
eBGP  VPC-194
EoMPLS
- Ethernet remote port shutdown  VPC-19
EoMPLS (Ethernet over MPLS)
- ethernet port mode  VPC-18
- Ethernet remote port shutdown  VPC-19
- inter-as port mode  VPC-20
- overview  VPC-18
- QinAny mode  VPC-22
- QinQ mode  VPC-21
- ethernet port mode  VPC-18
flooding
- MAC address  VPC-53
FVRF (front door virtual routing and forwarding)  VPC-146
Generic Routing Encapsulation (GRE over L3VPN)  VPC-211
Generic Routing Encapsulation Overview (L2VPN)  VPC-24
Inter-AS configurations
- BGP  VPC-201
- interprovider VPN  VPC-201
- L2VPN quality of service  VPC-32
- supported  VPC-201
- Inter-AS mode  VPC-20
interprovider VPN, MPLS VPN  VPC-201

IPSec
  L2TPv3  VPC-145
    mapping  VPC-146
    over L2TPv3  VPC-146
    scenarios  VPC-146
    securing tunnels  VPC-145
IPv4 loopback interface, how to configure  VPC-179
ISP requirements, MPLS L2VPN  VPC-17

L

L2TPv3
  benefits  VPC-141
  control message hashing  VPC-150
  dynamic
    configuring pseudowires  VPC-159
    sessions  VPC-142
IPSec  VPC-145
MPLS  VPC-141
multipoint tunnel network  VPC-171
operation  VPC-140
peer authentication  VPC-142
prerequisites  VPC-139
services  VPC-140
static sessions  VPC-142
xconnect support  VPC-141
L2VPN, QoS restrictions  VPC-32
limit, MAC address
  actions  VPC-54
  configuring  VPC-93

M

MAC address
  aging  VPC-54
  flooding  VPC-53
  forwarding  VPC-53
  limit actions  VPC-54
related parameters  VPC-53
source-based learning  VPC-54
withdrawal  VPC-55
MDRR/WRED  VPC-172
modified deficit round robin  VPC-172
MPLS L2VPN
  configuration examples  VPC-37
  configuring  VPC-26
  configuring QoS in VLAN mode  VPC-34
  high availability  VPC-23
  interface or connection, how to configure  VPC-26
  ISP requirements  VPC-17
  Quality of service (QoS)  VPC-22
MPLS Layer 3 VPN
  automatic route distinguisher  VPC-199
  autonomous system  VPC-200
  components  VPC-195
  concepts  VPC-195
  customer edge router  VPC-196
  customer router  VPC-196
  defined  VPC-195
  defining  VPC-195
  distributed routing information  VPC-197
  FIB  VPC-194
  implementing  VPC-195
  major components  VPC-199
  MPLS forwarding  VPC-198
  PE router  VPC-196
  prerequisites  VPC-194
  provider router  VPC-196
  restrictions  VPC-194
  scalability  VPC-196
  security  VPC-196
  topology  VPC-196
  VPN routing information  VPC-198
  working  VPC-197
MPLS VPN
  Inter-AS ASBRs  VPC-199
  major components  VPC-199
MPLS VPN over IP tunnel  VPC-186

VFI (virtual forwarding instance)
   adding under bridge domain  VPC-76
   associating bridge domain member  VPC-79
   associating pseudowires  VPC-77
   attaching pseudowire classes to pseudowires  VPC-81
   configuring AToM pseudowires  VPC-83
   disabling  VPC-85

VLAN
   figure, mode packet flow  VPC-20
   mode  VPC-19

VPLS (Virtual Private LAN Services)
   overview  VPC-47

VPLS (virtual private LAN services)
   attachment circuits  VPC-48
   defining bridge domain  VPC-53
   Layer 2 VPN, architecture  VPC-48
   simulating virtual bridge  VPC-48

VPN services
   L2TPv3  VPC-171
   MPLS  VPC-171

VRF (virtual routing and forwarding)
   configuring backbone carrier core  VPC-261
   configuring CSC-PE routers  VPC-186

W

withdrawal, MAC address
   defining  VPC-55
   fields  VPC-91
   how to define  VPC-55

X

xconnects
   support  VPC-141