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 the Cisco IOS XR Software* module in the *Cisco IOS XR System Management Configuration Guide for the Cisco CRS Router*.

Note For a complete description of the commands listed in this module, refer to the *Cisco IOS XR Virtual Private Network Command Reference for the Cisco CRS Router*. 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

Release	Modification
Release 3.3.0	This feature was introduced.
Release 3.4.0	Support was added for MPLS L3VPN Carrier Supporting Carrier (CSC) functionality, including conceptual information and configuration tasks.
Release 3.5.0	Support was added for 6VPE. MPLS L3VPN Carrier Supporting Carrier (CSC) information was upgraded.
Release 3.6.0	Support was added for Inter-AS and CSC over IP Tunnels.

Release	Modification
Release 3.7.0	Support was added for: • IPv6 VPN Provider Edge (6VPE). • Inter-AS support for 6VPE.
Release 3.9.0	Support for Generic Routing Encapsulation (GRE) was added.

Release	Modification
Release 3.7.2	This feature was introduced.
Release 4.2.0	Support for Generic Routing Encapsulation (GRE) was added on A9K-SIP-700 line card.
Release 4.2.1	The maximum number of supported tunnel interfaces was increased to 2000 for the ASR 9000 Enhanced Ethernet and ASR 9000 Ethernet line cards.

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

- To configure MPLS Layer 3 VPNs, routers must support MPLS forwarding and Forwarding Information Base (FIB).

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, for procedures)
- These 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

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 non adjacent routers.
- Inter-AS supports IPv4 routes only. IPv6 is not supported.

Note The physical interfaces that connect the BGP speakers must support FIB and MPLS.

The following restrictions apply to routing protocols OSPF and RIP:

- IPv6 is not supported on OSPF and RIP.

Information About MPLS Layer 3 VPNs

To implement MPLS Layer 3 VPNs, you need to understand the following concepts:

MPLS L3VPN Overview

Before defining an MPLS VPN, VPN in general must be defined. A VPN is:

- An IP-based network delivering private network services over a public infrastructure
- A set of sites that are allowed to communicate with each other privately over the Internet or other public or private networks

Conventional VPNs are created by configuring a full mesh of tunnels or permanent virtual circuits (PVCs) to all sites in a VPN. This type of VPN is not easy to maintain or expand, as adding a new site requires changing each edge device in the VPN.

MPLS-based VPNs are created in Layer 3 and are based on the peer model. The peer model enables the service provider and the customer to exchange Layer 3 routing information. The service provider relays the data between the customer sites without customer involvement.

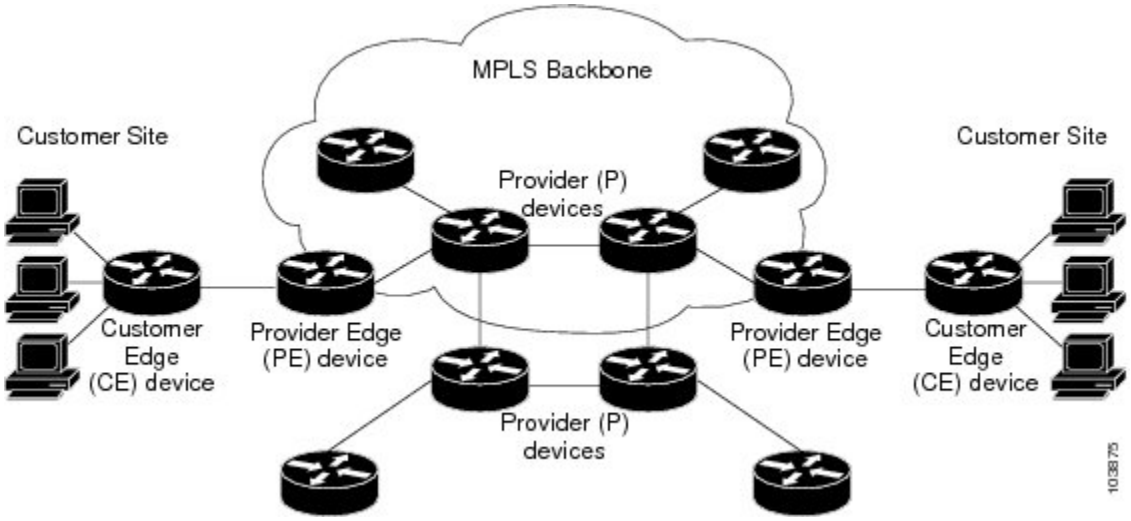
MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the edge router of the service provider that provides services to the customer site needs to be updated.

The components of the MPLS VPN are described as follows:

- Provider (P) router—Router in the core of the provider network. PE routers run MPLS switching and do not attach VPN labels to routed packets. VPN labels are used to direct data packets to the correct private network or customer edge router.
- PE router—Router that attaches the VPN label to incoming packets based on the interface or subinterface on which they are received, and also attaches the MPLS core labels. A PE router attaches directly to a CE router.
- Customer (C) router—Router in the Internet service provider (ISP) or enterprise network.
- Customer edge (CE) router—Edge router on the network of the ISP that connects to the PE router on the network. A CE router must interface with a PE router.

This following figures shows a basic MPLS VPN topology.

Figure 1: Basic MPLS VPN Topology



MPLS L3VPN Benefits

MPLS L3VPN provides the following benefits:

- Service providers can deploy scalable VPNs and deliver value-added services.
- Connectionless service guarantees that no prior action is necessary to establish communication between hosts.
- Centralized Service: Building VPNs in Layer 3 permits delivery of targeted services to a group of users represented by a VPN.
- Scalability: Create scalable VPNs using connection-oriented, point-to-point overlays, Frame Relay, or ATM virtual connections.
- Security: Security is provided at the edge of a provider network (ensuring that packets received from a customer are placed on the correct VPN) and in the backbone.
- Integrated Quality of Service (QoS) support: QoS provides the ability to address predictable performance and policy implementation and support for multiple levels of service in an MPLS VPN.
- Straightforward Migration: Service providers can deploy VPN services using a straightforward migration path.
- Migration for the end customer is simplified. There is no requirement to support MPLS on the CE router and no modifications are required for a customer intranet.

How MPLS L3VPN Works

MPLS VPN functionality is enabled at the edge of an MPLS network. The PE router performs the following tasks:

- Exchanges routing updates with the CE router
- Translates the CE routing information into VPN version 4 (VPNv4) and VPN version 6 (VPNv6) routes.
- Exchanges VPNv4 and VPNv6 routes with other PE routers through the Multiprotocol Border Gateway Protocol (MP-BGP)

Virtual Routing and Forwarding Tables

Each VPN is associated with one or more VPN routing and forwarding (VRF) instances. A VRF defines the VPN membership of a customer site attached to a PE router. A VRF consists of the following components:

- An IP version 4 (IPv4) unicast routing table
- A derived FIB table
- A set of interfaces that use the forwarding table
- A set of rules and routing protocol parameters that control the information that is included in the routing table

These components are collectively called a VRF instance.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A site can be a member of multiple VPNs. However, a site can associate with only one VRF. A VRF contains all the routes available to the site from the VPNs of which it is a member.

Packet forwarding information is stored in the IP routing table and the FIB table for each VRF. A separate set of routing and FIB tables is maintained for each VRF. These tables prevent information from being forwarded outside a VPN and also prevent packets that are outside a VPN from being forwarded to a router within the VPN.

VPN Routing Information: Distribution

The distribution of VPN routing information is controlled through the use of VPN route target communities, implemented by BGP extended communities. VPN routing information is distributed as follows:

- When a VPN route that is learned from a CE router is injected into a BGP, a list of VPN route target extended community attributes is associated with it. Typically, the list of route target community extended values is set from an export list of route targets associated with the VRF from which the route was learned.
- An import list of route target extended communities is associated with each VRF. The import list defines route target extended community attributes that a route must have for the route to be imported into the VRF. For example, if the import list for a particular VRF includes route target extended communities A, B, and C, then any VPN route that carries any of those route target extended communities—A, B, or C—is imported into the VRF.

BGP Distribution of VPN Routing Information

A PE router can learn an IP prefix from the following sources:

- A CE router by static configuration

- An eBGP session with the CE router
- A Routing Information Protocol (RIP) exchange with the CE router
- Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and RIP as Interior Gateway Protocols (IGPs)

The IP prefix is a member of the IPv4 address family. After the PE router learns the IP prefix, the PE converts it into the VPN-IPv4 prefix by combining it with a 64-bit route distinguisher. The generated prefix is a member of the VPN-IPv4 address family. It uniquely identifies the customer address, even if the customer site is using globally nonunique (unregistered private) IP addresses. The route distinguisher used to generate the VPN-IPv4 prefix is specified by the **rd** command associated with the VRF on the PE router.

BGP distributes reachability information for VPN-IPv4 prefixes for each VPN. BGP communication takes place at two levels:

- Within the IP domain, known as an autonomous system.
- Between autonomous systems.

PE to PE or PE to route reflector (RR) sessions are iBGP sessions, and PE to CE sessions are eBGP sessions. PE to CE eBGP sessions can be directly or indirectly connected (eBGP multihop).

BGP propagates reachability information for VPN-IPv4 prefixes among PE routers by the BGP protocol extensions (see RFC 2283, Multiprotocol Extensions for BGP-4), which define support for address families other than IPv4. Using the extensions ensures that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.

MPLS Forwarding

Based on routing information stored in the VRF IP routing table and the VRF FIB table, packets are forwarded to their destination using MPLS.

A PE router binds a label to each customer prefix learned from a CE router and includes the label in the network reachability information for the prefix that it advertises to other PE routers. When a PE router forwards a packet received from a CE router across the provider network, it labels the packet with the label learned from the destination PE router. When the destination PE router receives the labeled packet, it pops the label and uses it to direct the packet to the correct CE router. Label forwarding across the provider backbone is based on either dynamic label switching or traffic engineered paths. A customer data packet carries two levels of labels when traversing the backbone:

- The top label directs the packet to the correct PE router.
- The second label indicates how that PE router should forward the packet to the CE router.

More labels can be stacked if other features are enabled. For example, if traffic engineering (TE) tunnels with fast reroute (FRR) are enabled, the total number of labels imposed in the PE is four (Layer 3 VPN, Label Distribution Protocol (LDP), TE, and FRR).

Automatic Route Distinguisher Assignment

To take advantage of iBGP load balancing, every network VRF must be assigned a unique route distinguisher. VRF is require a route distinguisher for BGP to distinguish between potentially identical prefixes received from different VPNs.

With thousands of routers in a network each supporting multiple VRFs, configuration and management of route distinguishers across the network can present a problem. Cisco IOS XR software simplifies this process by assigning unique route distinguisher to VRFs using the **rd auto** command.

To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across theVRFs.

Finally, route distinguisher values are checkpointed so that route distinguisher assignment to VRF is persistent across failover or process restart. If an route distinguisher is explicitely configured for a VRF, this value is not overridden by the autoroute distinguisher.

MPLS L3VPN Major Components

An MPLS-based VPN network has three major components:

- VPN route target communities—A VPN route target community is a list of all members of a VPN community. VPN route targets need to be configured for each VPN community member.
- Multiprotocol BGP (MP-BGP) peering of the VPN community PE routers—MP-BGP propagates VRF reachability information to all members of a VPN community. MP-BGP peering needs to be configured in all PE routers within a VPN community.
- MPLS forwarding—MPLS transports all traffic between all VPN community members across a VPN service-provider network.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A given site can be a member of multiple VPNs. However, a site can associate with only one VRF. A customer-site VRF contains all the routes available to the site from the VPNs of which it is a member

Inter-AS Support for L3VPN

This section contains the following topics:

Inter-AS Restrictions

Inter-AS functionality is available using VPNv4 only. VPNv6 is not currently supported.

Inter-AS Support: Overview

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

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

An MPLS VPN Inter-AS provides the following benefits:

- Allows a VPN to cross more than one service provider backbone.

  Service providers, running separate autonomous systems, can jointly offer MPLS VPN services to the same end customer. A VPN can begin at one customer site and traverse different VPN service provider backbones before arriving at another site of the same customer. Previously, MPLS VPN could traverse only a single BGP autonomous system service provider backbone. This feature lets multiple autonomous systems form a continuous, seamless network between customer sites of a service provider.

- Allows a VPN to exist in different areas.

  A service provider can create a VPN in different geographic areas. Having all VPN traffic flow through one point (between the areas) allows for better rate control of network traffic between the areas.

- Allows confederations to optimize iBGP meshing.

  Internal Border Gateway Protocol (iBGP) meshing in an autonomous system is more organized and manageable. You can divide an autonomous system into multiple, separate subautonomous systems and then classify them into a single confederation. This capability lets a service provider offer MPLS VPNs across the confederation, as it supports the exchange of labeled VPN-IPv4 Network Layer Reachability Information (NLRI) between the subautonomous systems that form the confederation.

Inter-AS and ASBRs

Separate autonomous systems from different service providers can communicate by exchanging IPv4 NLRI and IPv6 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.

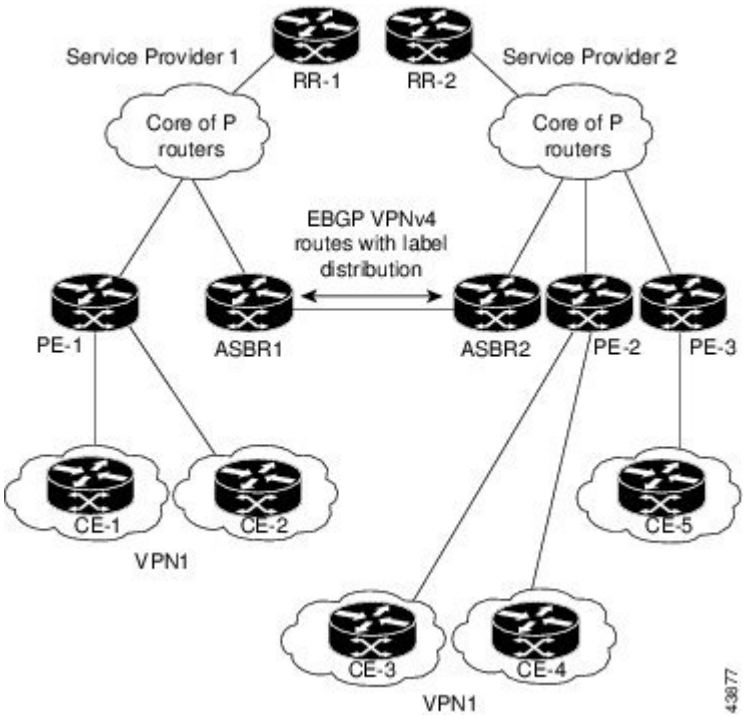
  Inter-AS configurations supported in an MPLS VPN can include:

  - 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

The following figure 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 2: eBGP Connection Between Two MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Addresses



This configuration uses the following process to transmit information:

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.
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.
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.)

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 confgured 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

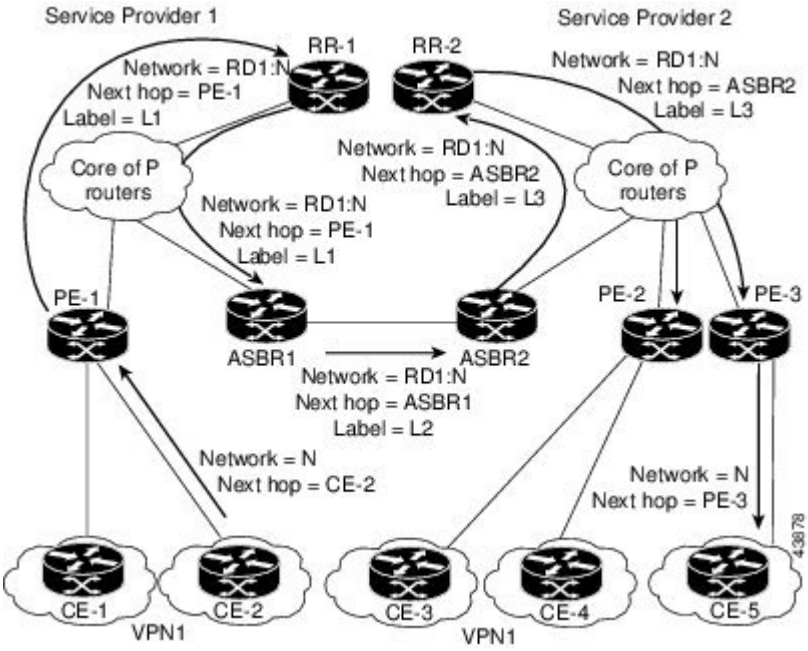
Autonomous systems exchange VPN routing information (routes and labels) to establish connections. To control connections between autonomous systems, the PE routers and eBGP border edge routers maintain a label forwarding information base (LFIB). The LFIB manages the labels and routes that the PE routers 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.

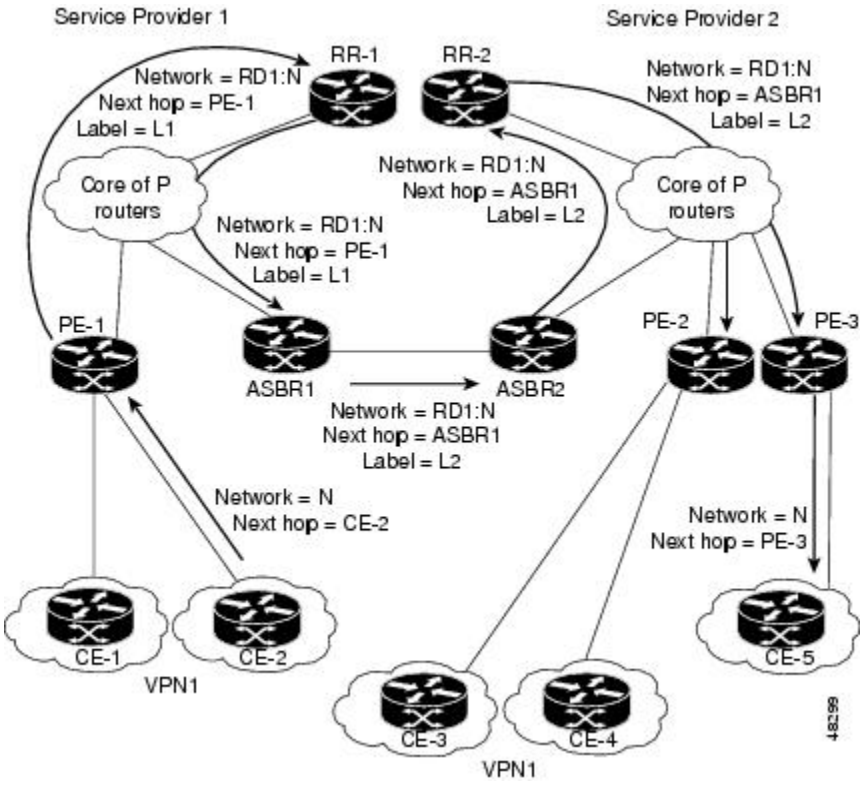
Figure 3: Exchanging Routes and Labels Between MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Address



The following figure 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 The following figure is not applicable to Inter-AS over IP tunnels.

Figure 4: Exchanging Routes and Labels with the redistributed Command in an MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses



Packet Forwarding

Note This section is not applicable to Inter-AS over IP tunnels.

The figure below 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.

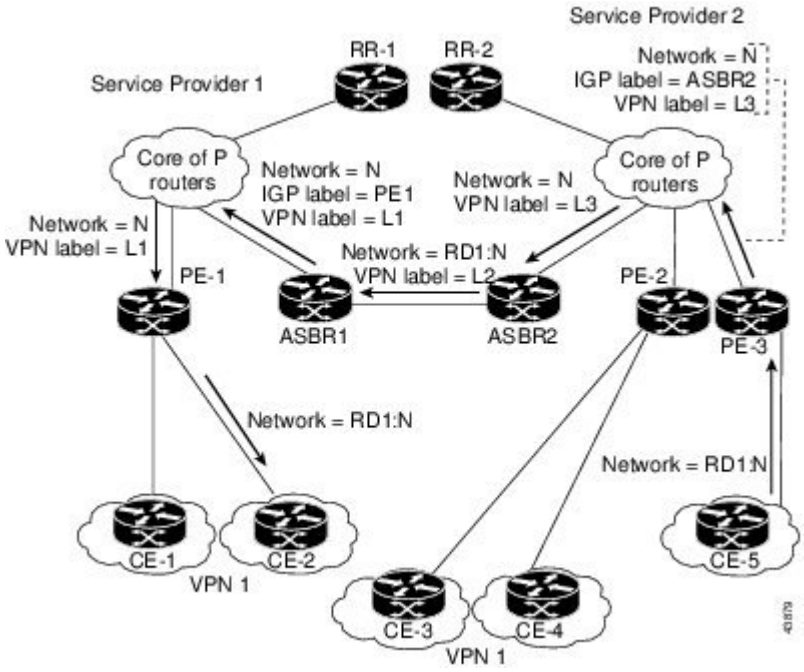
The service provider VPN backbone uses dynamic label switching to forward labels.

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:

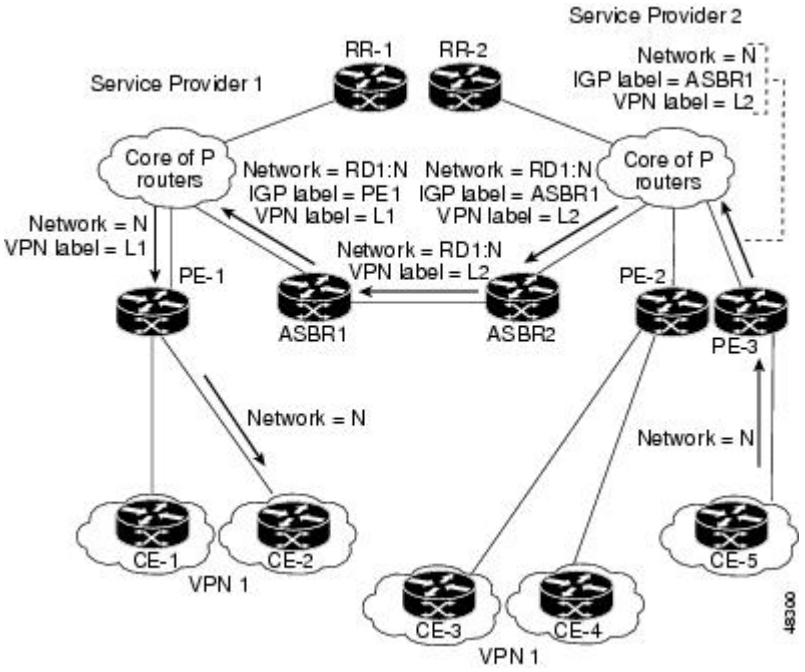
- The first label (IGP route label) directs the packet to the correct PE router on the eBGP border edge router. (For example, the IGP label of ASBR2 points to the ASBR2 border edge router.)
- The second label (VPN route label) directs the packet to the appropriate PE router or eBGP border edge router.

Figure 5: Forwarding Packets Between MPLS VPN Inter-AS Systems with ASBRs Exchanging VPN-IPv4 Addresses



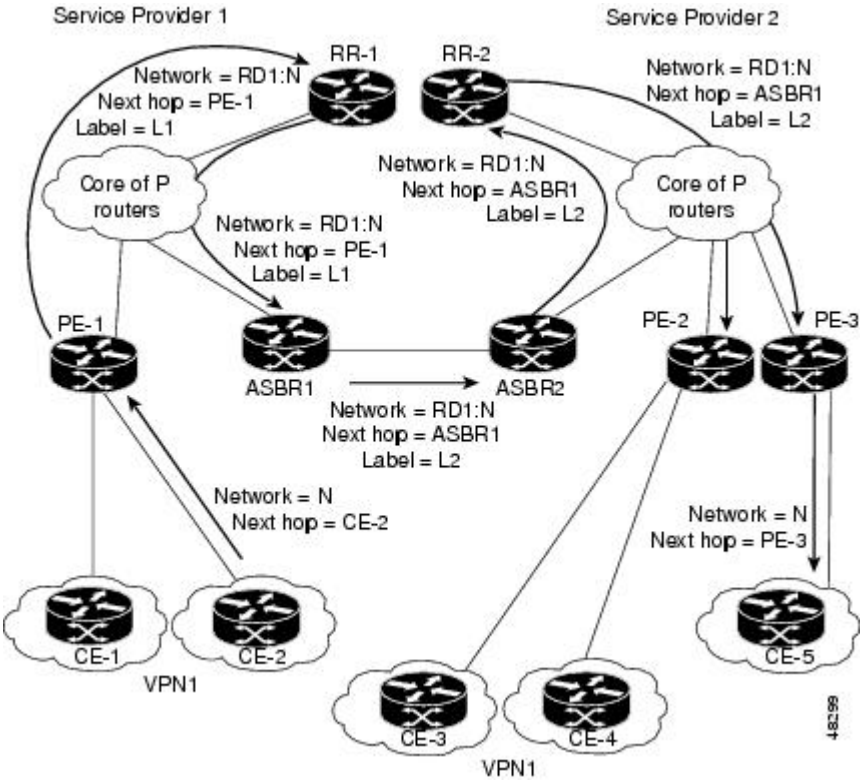
The following figure shows the same packet forwarding method, except the eBGP router (ASBR1) forwards the packet without reassigning a new label to it.

Figure 6: Forwarding Packets Without a New Label Assignment Between MPLS VPN Inter-AS System with ASBRs Exchanging VPN-IPv4 Addresses



The following figure illustrates the exchange of VPN route and label information between autonomous systems.

Figure 7: Exchanging Routes and Labels in an MPLS VPN Inter-AS with ASBRs



Confederations

A confederation is multiple subautonomous systems grouped together. A confederation reduces the total number of peer devices in an autonomous system. A confederation divides an autonomous system into subautonomous systems and assigns a confederation identifier to the autonomous systems. A VPN can span service providers running in separate autonomous systems or multiple subautonomous systems that form a confederation.

In a confederation, each subautonomous system is fully meshed with other subautonomous systems. The subautonomous systems communicate using an IGP, such as Open Shortest Path First (OSPF) or Intermediate System-to-Intermediate System (IS-IS). Each subautonomous system also has an eBGP connection to the other subautonomous systems. The confederation eBGP (CEBGP) border edge routers forward next-hop-self addresses between the specified subautonomous systems. The next-hop-self address forces the BGP to use a specified address as the next hop rather than letting the protocol choose the next hop.

You can configure a confederation with separate subautonomous systems two ways:

- Configure a router to forward next-hop-self addresses between only the CEBGP border edge routers (both directions). The subautonomous systems (iBGP peers) at the subautonomous system border do

not forward the next-hop-self address. Each subautonomous system runs as a single IGP domain. However, the CEBGP border edge router addresses are known in the IGP domains.

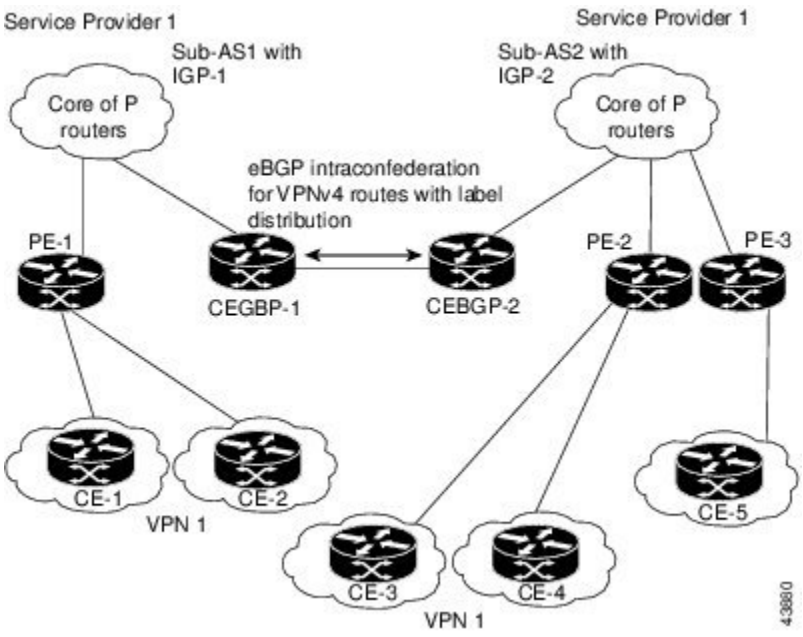
- Configure a router to forward next-hop-self addresses between the CEBGP border edge routers (both directions) and within the iBGP peers at the subautonomous system border. Each subautonomous system runs as a single IGP domain but also forwards next-hop-self addresses between the PE routers in the domain. The CEBGP border edge router addresses are known in the IGP domains.

Note eBGP Connection Between Two Subautonomous Systems in a Confederation figure 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.

The figure below 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 8: 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 .

MPLS VPN Inter-AS BGP Label Distribution

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up the MPLS VPN Inter-AS network so that the ASBRs exchange IPv4 routes with MPLS labels of the provider edge (PE) routers. Route reflectors (RRs) exchange VPN-IPv4 routes by using multihop, multiprotocol external Border Gateway Protocol (eBGP). This method of configuring the Inter-AS system is often called MPLS VPN Inter-AS BGP Label Distribution.

Configuring the Inter-AS system so that the ASBRs exchange the IPv4 routes and MPLS labels has the following benefits:

- Saves the ASBRs from having to store all the VPN-IPv4 routes. Using the route reflectors to store the VPN-IPv4 routes and forward them to the PE routers results in improved scalability compared with configurations in which the ASBR holds all the VPN-IPv4 routes and forwards the routes based on VPN-IPv4 labels.
- Having the route reflectors hold the VPN-IPv4 routes also simplifies the configuration at the border of the network.
- Enables a non-VPN core network to act as a transit network for VPN traffic. You can transport IPv4 routes with MPLS labels over a non-MPLS VPN service provider.
- Eliminates the need for any other label distribution protocol between adjacent label switch routers (LSRs). If two adjacent LSRs are also BGP peers, BGP can handle the distribution of the MPLS labels. No other label distribution protocol is needed between the two LSRs.

Exchanging IPv4 Routes with MPLS labels

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up a VPN service provider network to exchange IPv4 routes with MPLS labels. You can configure the VPN service provider network as follows:

- Route reflectors exchange VPN-IPv4 routes by using multihop, multiprotocol eBGP. This configuration also preserves the next-hop information and the VPN labels across the autonomous systems.

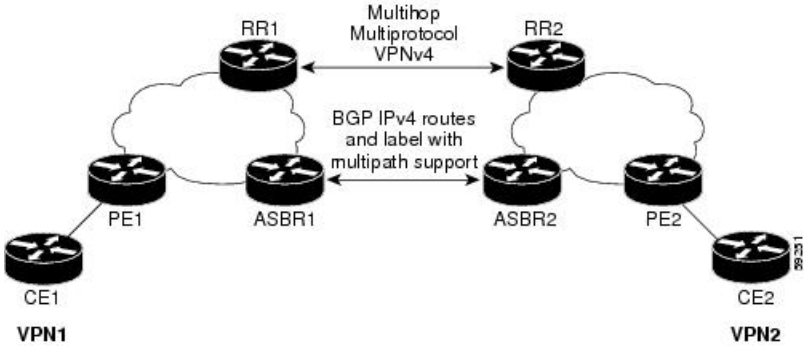
- A local PE router (for example, PE1 in the figure below) needs to know the routes and label information for the remote PE router (PE2).

  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 9: 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.
- VPNv6 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

- CSC Benefits
- Configuration Options for the Backbone and Customer Carriers

Throughout this document, the following terminology is used in the context of CSC:

backbone carrier—Service provider that provides the segment of the backbone network to the other provider. A backbone carrier offers BGP and MPLS VPN services.

customer carrier—Service provider that uses the segment of the backbone network. The customer carrier may be an Internet service provider (ISP) or a BGP/MPLS VPN service provider.

CE router—A customer edge router is part of a customer network and interfaces to a provider edge (PE) router. In this document, the CE router sits on the edge of the customer carrier network.

PE router—A provider edge router is part of a service provider's network connected to a customer edge (CE) router. In this document, the PE router sits on the edge of the backbone carrier network

ASBR—An autonomous system boundary router connects one autonomous system to another.

CSC Prerequisites

The following prerequisites are required to configure CSC:

- You must be able to configure MPLS VPNs with end-to-end (CE-to-CE router) pings working.
- You must be able to configure Interior Gateway Protocols (IGPs), MPLS Label Distribution Protocol (LDP), and Multiprotocol Border Gateway Protocol (MP-BGP).
- You must ensure that CSC-PE and CSC-CE routers support BGP label distribution.

Note BGP is the only supported label distribution protocol on the link between CE and PE.

CSC Benefits

This section describes the benefits of CSC to the backbone carrier and customer carriers.

Benefits to the Backbone Carrier

- The backbone carrier can accommodate many customer carriers and give them access to its backbone.
- The MPLS VPN carrier supporting carrier feature is scalable.
- The MPLS VPN carrier supporting carrier feature is a flexible solution.

Benefits to the Customer Carriers

- The MPLS VPN carrier supporting carrier feature removes from the customer carrier the burden of configuring, operating, and maintaining its own backbone.
- Customer carriers who use the VPN services provided by the backbone carrier receive the same level of security that Frame Relay or ATM-based VPNs provide.

- Customer carriers can use any link layer technology to connect the CE routers to the PE routers and the PE routers to the P routers.
- The customer carrier can use any addressing scheme and still be supported by a backbone carrier.

Benefits of Implementing MPLS VPN CSC Using BGP

The benefits of using BGP to distribute IPv4 routes and MPLS label routes are:

- BGP takes the place of an IGP and LDP in a VPN forwarding and routing instance (VRF) table.
- BGP is the preferred routing protocol for connecting two ISPs.

Configuration Options for the Backbone and Customer Carriers

To enable CSC, the backbone and customer carriers must be configured accordingly:

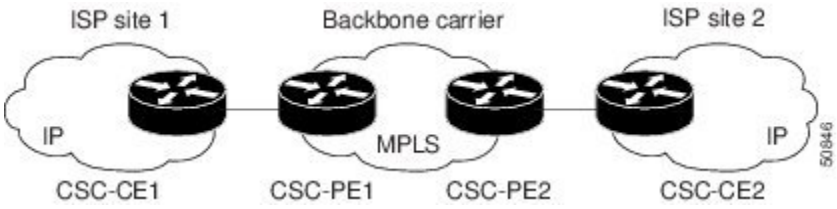
- The backbone carrier must offer BGP and MPLS VPN services.
- The customer carrier can take several networking forms. The customer carrier can be:
  - An ISP with an IP core (see the "Customer Carrier: ISP with IP Core").
  - An MPLS service provider with or without VPN services (see "Customer Carrier: MPLS Service Provider").

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

The following figure 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 10: Network: Customer Carrier Is an ISP

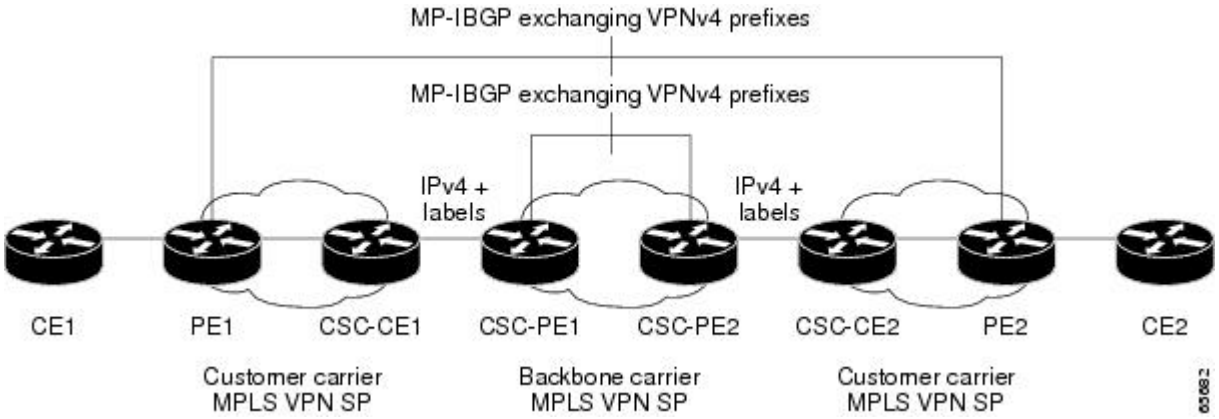


The links between the CE and PE routers use eBGP to distribute IPv4 routes and MPLS labels. Between the links, the PE routers use multiprotocol iBGP to distribute VPNv4 routes.

Customer Carrier: MPLS Service Provider

The following figure 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 11: Network: Customer Carrier Is an MPLS VPN Service Provider



In Network: Customer Carrier Is an MPLS VPN Service Provider configuration, 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”.

This section includes the follow subsections:

6VPE Benefits

6VPE provides the following benefits to service providers:

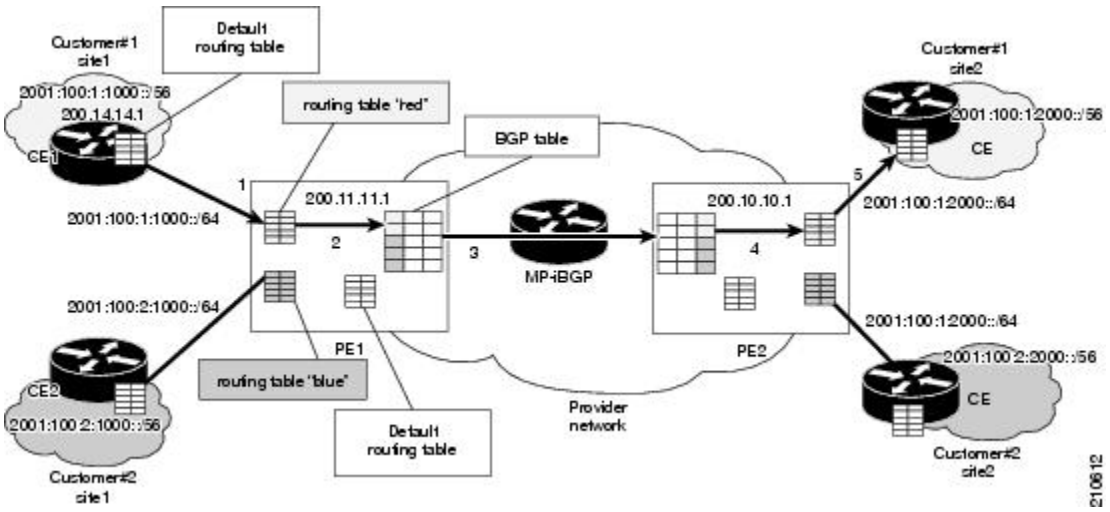
- Support for IPv6 without changing the IPv4 MPLS backbone.
- No requirement for a separate signaling plane.

- 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

The following figure illustrates the 6VPE network architecture and control plane protocols when two IPv6 sites communicate through an MPLSv4 backbone.

Figure 12: 6VPE Network Architecture



Dual Stack

Dual stack is a technique that lets IPv4 and IPv6 coexist on the same interfaces. Coexistence of IPv4 and IPv6 is a requirement for initial deployment. With regard to supporting IPv6 on a MPLS network, two important aspects of the network should be reviewed:

- *Core*: The 6VPE technique carries IPv6 in a VPN fashion over a non-IPv6-aware MPLS core, and enables IPv4 or IPv6 communities to communicate with each other over an IPv4 MPLS backbone without modifying the core infrastructure. By avoiding dual stacking on the core routers, the resources can be dedicated to their primary function to avoid any complexity on the operational side. The transition and integration with respect to the current state of networks is also transparent.
- *Access*: To support native IPv6, the access that connects to IPv4 and IPv6 domains must be IPv6-aware. Service provider edge elements can exchange routing information with end users; therefore, dual stacking is a mandatory requirement on the access layer.

6VPE Operation

When IPv6 is enabled on the subinterface that is participating in a VPN, it becomes an IPv6 VPN. The customer edge-provider edge link is running IPv6 or IPv4 natively. The addition of IPv6 on a provider edge router turns the provider edge into 6VPE, thereby enabling service providers to support IPv6 over the MPLS network.

Provider edge routers use VRF tables to maintain the segregated reachability and forwarding information of each IPv6 VPN. MPBGP with its IPv6 extensions distributes the routes from 6VPE to other 6VPEs through a direct IBGP session or through VPNv6 route reflectors. The next hop of the advertising provider edge router still remains the IPv4 address (normally it is a loopback interface), but with the addition of IPv6, a value of ::FFFF: is prepended to the IPv4 next hop.

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

Configuring the core network includes the following tasks:

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

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 .

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* chapter in the *Cisco IOS XR MPLS Configuration Guide for the Cisco CRS Router* for configuration information.
- MPLS Traffic Engineering Resource Reservation Protocol (RSVP)—See *Implementing RSVP for MPLS-TE and MPLS O-UNI* module in the *Cisco IOS XR MPLS Configuration Guide for the Cisco CRS Router* 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 module in the *Cisco IOS XR IP Addresses and Services Configuration Guide for the Cisco CRS Router*.

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family vpnv4 unicast** or **address-family vpnv6 unicast**

Example:

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

Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 4 **neighbor** *ip-address* **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24 remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **address-family vpnv4 unicast** or **address-family vpnv6 unicast**

Example:

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

Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 6 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

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

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

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

Enters VRF address family configuration mode for the IPv4 address family.

Step 4 **import route-policy** *policy-name*

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# import route-policy policy_A
```

Specifies a route policy that can be imported into the local VPN.

Step 5 **import route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 120:1
```

Allows exported VPN routes to be imported into the VPN if one of the route targets of the exported route matches one of the local VPN import route targets.

Step 6 **export route-policy** *policy-name*

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# export route-policy policy_B
```

Specifies a route policy that can be exported from the local VPN.

Step 7 **export route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 120:2
```

Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 8 **exit**

Example:

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

Exits VRF address family configuration mode and returns the router to VRF configuration mode.

Step 9 **exit**

Example:

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

Exits VRF configuration mode and returns the router to Global Configuration mode.

Step 10 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 11 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode for BGP routing.

Step 12 **rd** { *as-number* | *ip-address* | **auto** }

Example:

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

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

Step 13 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

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

Example:

```
RP/0/RP0/CPU0:router(config)# interface TenGigE 0/3/0/0
```

Enters interface configuration mode.

Step 3 **vrf** *vrf-name*

Example:

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

Configures a VRF instance and enters VRF configuration mode.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0
```

Configures a primary IPv4 address for the specified interface.

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

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

end - Prompts user to take one of these actions:

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

Configuring BGP 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. Do one of the following:
   - **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute isis** *process-id* [**level** {**1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2** | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **bgp router-id** {*ip-address*}

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24
```

Configures the local router with a router ID of 192.168.70.24.

Step 4 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for BGP routing.

Step 5 **label-allocation-mode per-ce**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce
```

Sets the MPLS VPN label allocation mode for each customer edge (CE) label mode allowing the provider edge (PE) router to allocate one label for every immediate next-hop.

Step 6 **address-family ipv4 unicast**

Example:

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

Enters VRF address family configuration mode for the IPv4 address family.

Step 7 Do one of the following:

- **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute isis** *process-id* [**level** {**1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospfv3** *process-id* [**match** { **external** [**1** | **2** | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]

Example:

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

Causes routes to be redistributed into BGP. The routes that can be redistributed into BGP are:

- Connected
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Static

Step 8 **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]

Example:

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

Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized.

- The **as-set** keyword generates autonomous system set path information and community information from contributing paths.
- The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths.
- The **summary-only** keyword filters all more specific routes from updates.
- The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.

Step 9 **network** {*ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]

Example:

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

Configures the local router to originate and advertise the specified network.

Step 10 **exit**

Example:

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

Exits VRF address family configuration mode and returns the router to VRF configuration mode for BGP routing.

Step 11 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24
```

Places the router in VRF neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 12 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 13 **password** { **clear** | **encrypted** } *password*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# password clear pswd123
```

Configures neighbor 172.168.40.24 to use MD5 authentication with the password pswd123.

Step 14 **ebgp-multihop** [*ttl-value*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop
```

Allows a BGP connection to neighbor 172.168.40.24.

Step 15 **address-family ipv4 unicast**

Example:

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

Enters VRF neighbor address family configuration mode for BGP routing.

Step 16 **allowas-in** [*as-occurrence-number*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 3
```

Replaces the neighbor autonomous system number (ASN) with the PE ASN in the AS path three times.

Step 17 **route-policy** *route-policy-name* **in**

Example:

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

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

Step 18 **route-policy** *route-policy-name* **out**

Example:

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

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

Step 19 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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. Do one of the following:
   - **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
   - **redistribute connected** [**route-policy** *name*]
   - **redistribute isis** *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
   - **redistribute eigrp** *as-number* [**route-policy** *name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**route-policy** *name*]
   - **redistribute static** [**route-policy** *name*]
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router rip**

Example:

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

Enters the Routing Information Protocol (RIP) configuration mode allowing you to configure the RIP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-rip)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for RIP routing.

Step 4 **interface** *type instance*

Example:

```
RP/0/RP0/CPU0:router(config-rip-vrf)# interface TenGigE 0/3/0/0
```

Enters VRF interface configuration mode.

Step 5 **site-of-origin** { *as-number : number* | *ip-address : number* }

Example:

```
RP/0/RP0/CPU0:router(config-rip-vrf-if)# site-of-origin 200:1
```

Identifies routes that have originated from a site so that the re-advertisement of that prefix back to the source site can be prevented. Uniquely identifies the site from which a PE router has learned a route.

Step 6 **exit**

Example:

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

Exits VRF interface configuration mode, and returns the router to VRF configuration mode for RIP routing.

Step 7 Do one of the following:

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

- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**route-policy** *name*]
- **redistribute static** [**route-policy** *name*]

Example:

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

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

Step 8 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring Static 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router static**

Example:

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

Enters static routing configuration mode allowing you to configure the static routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-static)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for static routing.

Step 4 **address-family ipv4 unicast**

Example:

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

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 *prefix/mask* [**vrf** *vrf-name*] { *ip-address* | *type interface-path-id* }

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 10.1.1.1
```

Assigns the static route to vrf_1.

Step 6 *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**

Example:

```
RP/0/RP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 bfd fast-detect
```

Enables bidirectional forwarding detection (BFD) to detect failures in the path between adjacent forwarding engines.

This option is available is when the forwarding router address is specified in Step 5 .

Step 7 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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. Do one of the following:
   - **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **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*]
   - **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **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*]
   - **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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

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

Enters OSPF configuration mode allowing you to configure the OSPF routing process.

Step 3 **vrf** *vrf-name*

Example:

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

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4 **router-id** {*router-id* | type interface-path-id}

Example:

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

Configures the router ID for the OSPF routing process.

Step 5 Do one of the following:

- **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **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*]
- **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **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*]
- **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

Example:

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

Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are:

- Border Gateway Protocol (BGP)
- Connected
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- OSPF
- Static
- Routing Information Protocol (RIP)

Step 6 **area** *area-id*

Example:

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

Configures the OSPF area as area 0.

Step 7 **interface** type interface-path-id

Example:

```
RP/0/RP0/CPU0:router(config-ospf-vrf-ar)# interface TenGigE 0/3/0/0
```

Associates interface TenGigE 0/3/0/0 with area 0.

Step 8 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring EIGRP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Enhanced Interior Gateway Routing Protocol (EIGRP).

Using EIGRP between the PE and CE routers allows you to transparently connect EIGRP customer networks through an MPLS-enable Border Gateway Protocol (BGP) core network so that EIGRP routes are redistributed through the VPN across the BGP network as internal BGP (iBGP) routes.

Before You Begin

BGP is configured in the network. See the *Implementing BGP* module in the *Cisco IOS XR Routing Configuration Guide for the Cisco CRS Router*

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router eigrp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router eigrp 24
```

Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for EIGRP routing.

Step 4 **address-family ipv4**

Example:

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

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 **router-id** *router-id*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# router-id 172.20.0.0
```

Configures the router ID for the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process.

Step 6 **autonomous-system** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 6
```

Configures the EIGRP routing process to run within a VRF.

Step 7 **default-metric** *bandwidth delay reliability loading mtu*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# default-metric 100000 4000 200 45 4470
```

Sets the metrics for an EIGRP.

Step 8 **redistribute** { { **bgp** | **connected** | **isis** | **ospf** | **rip** | **static** } [*as-number* | *instance-name*] } [**route-policy** *name*]

Example:

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

Causes connected routes to be redistributed into EIGRP.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# interface TenGigE 0/3/0/0
```

Associates interface TenGigE 0/3/0/0 with the EIGRP routing process.

Step 10 **site-of-origin** { *as-number:number* | *ip-address : number* }

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 201:1
```

Configures site of origin (SoO) on interface TenGigE 0/3/0/0.

Step 11 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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.

Before You Begin

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*module in the *Cisco IOS XR Routing Configuration Guide for the Cisco CRS Router*.

Restriction 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router eigrp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router eigrp 24
```

Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-eigrp)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode for EIGRP routing.

Step 4 **address-family ipv4**

Example:

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

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]

Example:

```
RP/0/RP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 24 route-policy policy_A
```

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

Step 6 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

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

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

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
RP/0/RP0/CPU0:router(config-bgp)#
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)#
```

Enters global address family configuration mode for the IPv4 unicast address family.

Step 4 **allocate-label all**

Example:

```
RP/0/CPU0:router(config-bgp-af)# allocate-label all
```

Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24

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

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 6 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 7 **address-family ipv4 labeled-unicast**

Example:

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

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

Step 8 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies a routing policy to updates that are received from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **in** keyword to define the policy for inbound routes.

Step 9 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies a routing policy to updates that are sent to a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **out** keyword to define the policy for outbound routes.

Step 10 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring the Route Reflectors to Exchange VPN-IPv4 Routes

Perform this task to enable the route reflectors to exchange VPN-IPv4 routes by using multihop. This task specifies that the next-hop information and the VPN label are to be preserved across the autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **ebgp-multihop** [*ttl-value*]
6. **update-source** *type interface-path-id*
7. **address-family vpnv4 unicast**
8. **route-policy** *route-policy-name* **in**
9. **route-policy** *route-policy-name* **out**
10. **next-hop-unchanged**
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
RP/0/RP0/CPU0:router(config-bgp)#
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 172.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **ebgp-multihop** [*ttl-value*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# ebgp-multihop
```

Enables multihop peerings with external BGP neighbors.

Step 6 **update-source** *type interface-path-id*

Example:

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

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 7 **address-family vpnv4 unicast**

Example:

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

Configures VPNv4 address family.

Step 8 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies a routing policy to updates that are received from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **in** keyword to define the policy for inbound routes.

Step 9 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies a routing policy to updates that are sent to a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **out** keyword to define the policy for outbound routes.

Step 10 **next-hop-unchanged**

Example:

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

Disables overwriting of the next hop before advertising to external Border Gateway Protocol (eBGP) peers.

Step 11 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring the 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-bgp-af)#
```

Enters global address family configuration mode for the IPv4 unicast address family.

Step 4 **allocate-label all**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# allocate-label all
```

Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 7 **update-source** *type interface-path-id*

Example:

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

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 8 **address-family ipv4 labeled-unicast**

Example:

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

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

Step 9 **route-reflector-client**

Example:

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

Configures the router as a BGP route reflector and neighbor 172.168.40.24 as its client.

Step 10 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# neighbor 10.40.25.2
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address .40.25.2 as an VPNv4 iBGP peer.

Step 11 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

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

Example:

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

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 13 **address-family ipv4 labeled-unicast**

Example:

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

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

Step 14 **route-reflector-client**

Example:

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

Configures the neighbor as a route reflector client.

Step 15 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

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 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.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** { **ipv4 tunnel** }
4. **address-family** { **vpnv4 unicast** }
5. **retain route-target** { **all** | **route-policy** *route-policy-name* }
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. **neighbor** *ip-address*
12. **remote-as** *autonomous-system-number*
13. **update-source** *type interface-path-id*
14. **address-family** { **ipv4 tunnel** }
15. **address-family** { **vpnv4 unicast** }
16. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

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

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:
```
RP/0/RP0/CPU0:router(config)# router bgp 120
RP/0/RP0/CPU0:router(config-bgp)#
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family** { **ipv4 tunnel** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# address-family ipv4 tunnel
RP/0/RP0/CPU0:router(config-bgp-af)#
```

Configures IPv4 tunnel address family.

Step 4 **address-family** { **vpnv4 unicast** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-af)# address-family vpnv4 unicast
```
Configures VPNv4 address family.

Step 5 **retain route-target** { **all** | **route-policy** *route-policy-name* }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-af)# retain route-target route-policy policy1
```

Retrieves VPNv4 table from PE routers.

The **retain route-target** command is required on an Inter-AS option B ASBR. You can use this command with either **all** or **route-policy** keyword

Step 6 **neighbor** *ip-address*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 7 **remote-as** *autonomous-system-number*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 8 **address-family** { **vpnv4 unicast** }

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

Configures VPNv4 address family.

Step 9 **route-policy** *route-policy-name* { **in** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies a routing policy to updates that are received from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **in** keyword to define the policy for inbound routes.

Step 10 **route-policy** *route-policy-name* { **out** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies a routing policy to updates that are sent from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the route policy. The example shows that the route policy name is defined as pass-all.
- Use the **out** keyword to define the policy for outbound routes.

Step 11 **neighbor** *ip-address*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# neighbor 175.40.25.2
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 175.40.25.2 as an VPNv4 iBGP peer.

Step 12 **remote-as** *autonomous-system-number*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 13 **update-source** *type interface-path-id*

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

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 14 **address-family** { **ipv4 tunnel** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family ipv4 tunnel
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#
```

Configures IPv4 tunnel address family.

Step 15 **address-family** { **vpnv4 unicast** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# address-family vpnv4 unicast
```
Configures VPNv4 address family.

Step 16 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

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

Enters the Global Configuration mode.

Step 2 **router static**

Example:

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

Enters router static configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-static-afi)#
```

Enables an IPv4 address family.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9
```

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

Step 5 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
RP/0/RP0/CPU0:router(config-bgp)#
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **bgp confederation peers** *peer autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# bgp confederation peers 8
```

Configures the peer autonomous system number that belongs to the confederation.

Step 4 **bgp confederation identifier** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# bgp confederation identifier 5
```

Specifies the autonomous system number for the confederation ID.

Step 5 **address-family vpnv4 unicast**

Example:

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

Configures VPNv4 address family.

Step 6 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)# neighbor 10.168.40.24
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 10.168.40.24 as a BGP peer.

Step 7 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 8 **address-family vpnv4 unicast**

Example:

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

Configures VPNv4 address family.

Step 9 **route-policy** *route-policy-name* **in**

Example:

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

Applies a routing policy to updates received from a BGP neighbor.

Step 10 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# route-policy Out-Ipv4 out
```

Applies a routing policy to updates advertised to a BGP neighbor.

Step 11 **next-hop-self**

Example:

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

Disables next-hop calculation and let you insert your own address in the next-hop field of BGP updates.

Step 12 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
RP/0/RP0/CPU0:router(config-bgp)
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **mpls activate**

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# mpls activate
RP/0/RP0/CPU0:router(config-bgp-mpls)#
```

Enters BGP MPLS activate configuration mode.

Step 4 **interface** *type interface-path-id*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-mpls)# interface GigabitEthernet 0/3/0/0
```

Enables MPLS on the interface.

Step 5 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router static**

Example:

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

Enters router static configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-static)# address-family ipv4 unicast
RP/0/RP0/CPU0:router(config-static-afi)#
```

Enables an IPv4 address family.

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

Example:

```
RP/0/RP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9
```

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

Step 5 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring Carrier Supporting Carrier

Perform the tasks in this section to configure Carrier Supporting Carrier (CSC):

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

Step 1 Identify the type of customer carrier, ISP, or MPLS VPN service provider.
Sets up requirements for configuration of carrier supporting carrier network.

Step 2 Identify the CE routers.
Sets up requirements for configuration of CE to PE connections.

Step 3 Identify the customer carrier core router configuration.
Sets up requirements for configuration between core (P) routers and between P routers and edge routers (PE and CSC-CE routers).

Step 4 Identify the customer carrier edge (CSC-CE) routers.
Sets up requirements for configuration of CSC-CE to CSC-PE connections.

Step 5 Identify the backbone carrier router configuration.
Sets up requirements for configuration between CSC core routers and between CSC core routers and edge routers (CSC-CE and CSC-PE routers).

Configuring the Backbone Carrier Core

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

- Verify IP connectivity in the CSC core.
- Verify LDP configuration in the CSC core.

Note This task is not applicable to CSC over IP tunnels.

- Configure VRFs for CSC-PE routers.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

Configuring the CSC-PE and CSC-CE Routers

Perform the following tasks to configure links between a CSC-PE router and the carrier CSC-CE router for an MPLS VPN CSC network that uses BGP to distribute routes and MPLS labels:

The following figure 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 13: 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 2
```

Configures a BGP routing process and enters router configuration mode.

- Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3 **address-family vpnv4 unicast**

Example:

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

Configures VPNv4 address family.

Step 4 **neighbor** *A.B.C.D*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-af)#neighbor 10.10.10.0
RP/0/RP0/CPU0:router(config-bgp-nbr)#
```

Configures the IP address for the BGP neighbor.

Step 5 **remote-as** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 888
```

Configures the AS number for the BGP neighbor.

Step 6 **update-source** *type interface-path-id*

Example:

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

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 7 **address-family vpnv4 unicast**

Example:

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

Configures VPNv4 unicast address family.

Step 8 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)# vrf 9999
RP/0/RP0/CPU0:router(config-bgp-vrf)#
```

Configures a VRF instance.

Step 9 **rd** {*as-number:nn* | *ip-address:nn* | **auto**}

Example:

```
RP/0/RP0/CPU0:router(onfig-bgp-vrf)# rd auto
RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
```

Configures a route distinguisher.

Note Use the **auto** keyword to automatically assign a unique route distinguisher.

Step 10 **address-family ipv4 unicast**

Example:

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

Example:
Configures IPv4 unicast address family.

Step 11 **allocate-label all**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# allocate-label all
```

Allocate labels for all local prefixes and prefixes received with labels.

Step 12 **neighbor** *A.B.C.D*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-af)# neighbor 10.10.10.0
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)#
```

Configures the IP address for the BGP neighbor.

Step 13 **remote-as** *as-number*

Example:
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)#remote-as 888
Enables the exchange of information with a neighboring BGP router.

Step 14 **address-family ipv4 labeled-unicast**

Example:

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

Configures IPv4 labeled-unicast address family.

Step 15 **route-policy** *route-policy-name* **in**

Example:

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

Applies the pass-all policy to all inbound routes.

Step 16 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out
```

Applies the pass-all policy to all outbound routes.

Step 17 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring PE to CE Core

This task describes how to configure a PE to Customer Edge (CE) core.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **bgp router-id** *ip-address*
5. **label-allocation-mode { per-ce | per-vrf }**
6. **address-family ipv6 unicast**
7. **redistribute {connected | static | eigrp }**
8. **neighbor** *ip-address*
9. **remote-as** *as-number*
10. **ebgp-multihop {** *maximum hops* **| mpls }**
11. **address-family ipv6 unicast**
12. **site-of-origin** [*as-number:nn* | *ip-address:nn*]
13. **as-override**
14. **allowas-in [** *as-occurrence-number***]**
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

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

Enters the Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 10
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **vrf** *vrf-name*

Example:

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

Configures a VRF instance.

Step 4 **bgp router-id** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)#bgp router-id 172.16.9.9
```

Configures a fixed router ID for a BGP-speaking router.

Step 5 **label-allocation-mode { per-ce | per-vrf }**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce
```

Configures the per-CE label allocation mode to avoid an extra lookup on the PE router and conserve label space (per-prefix is the default label allocation mode). In this mode, the PE router allocates one label for every immediate next-hop (in most cases, this would be a CE router). This label is directly mapped to the next hop, so there is no VRF route lookup performed during data forwarding. However, the number of labels allocated would be one for each CE rather than one for each VRF. Because BGP knows all the next hops, it assigns a label for each next hop (not for each PE-CE interface). When the outgoing interface is a multiaccess interface and the media access control (MAC) address of the neighbor is not known, Address Resolution Protocol (ARP) is triggered during packet forwarding.

The per-vrf keyword configures the same label to be used for all the routes advertised from a unique VRF.

Step 6 **address-family ipv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast
```

Specifies an IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 7 **redistribute {connected | static | eigrp }**

Example:

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

Causes routes from the specified instance to be redistributed into BGP.

Step 8 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 10.0.0.0
```

Configures a CE neighbor. The ip-address argument must be a private address.

Step 9 **remote-as** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2
```

Configures the remote AS for the CE neighbor.

Step 10 **ebgp-multihop {** *maximum hops* **| mpls }**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop 55
```

Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected.

Step 11 **address-family ipv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv6 unicast
```

Specifies an IPv6 address family unicast and enters address family configuration submode.

To see a list of all the possible keywords and arguments for this command, use the CLI help (?).

Step 12 **site-of-origin** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# site-of-origin 234:111
```

Configures the site-of-origin (SoO) extended community. Routes that are learned from this CE neighbor are tagged with the SoO extended community before being advertised to the rest of the PEs. SoO is frequently used to detect loops when as-override is configured on the PE router. If the prefix is looped back to the same site, the PE detects this and does not send the update to the CE.

Step 13 **as-override**

Example:

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

Configures AS override on the PE router. This causes the PE router to replace the CE's ASN with its own (PE) ASN.

Note This loss of information could lead to routing loops; to avoid loops caused by as-override, use it in conjunction with site-of-origin.

Step 14 **allowas-in [** *as-occurrence-number*]

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 5
```

Allows an AS path with the PE autonomous system number (ASN) a specified number of times.

Hub and spoke VPN networks need the looping back of routing information to the HUB PE through the HUB CE. When this happens, due to the presence of the PE ASN, the looped-back information is dropped by the HUB PE. To avoid this, use the allowas-in command to allow prefixes even if they have the PEs ASN up to the specified number of times.

Step 15 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring a Static Route to a Peer

Perform this task to configure a static route to an Inter-AS or CSC-CE peer.

When you configure an Inter-AS or CSC peer, BGP allocates a label for a /32 route to that peer and performs a NULL label rewrite. When forwarding a labeled packet to the peer, the router removes the top label from the label stack; however, in such an instance, BGP expects a /32 route to the peer. This task ensures that there is, in fact, a /32 route to the peer.

Please be aware of the following facts before performing this task:

- A /32 route is not required to establish BGP peering. A route using a shorter prefix length will also work.
- A shorter prefix length route is not associated with the allocated label; even though the BGP session comes up between the peers, without the static route, forwarding will not work.

Note To configure a static route on a CSC-PE, you must configure the router under the VRF (as noted in the detailed steps).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **router static**

Example:

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

Enters router static configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

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

Enables an IPv4 address family.

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

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

Example:

```
RP/0/RP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9
```

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

Step 5 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Verifying 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 vrf** *vrf-name*
15. **show bgp vrf** *vrf-name* **imported-routes**
16. **show route vrf** *vrf-name ip-address*
17. **show cef vrf** *vrf-name ip-address*
18. **show cef vrf** *vrf-name ip-address* **location** *node-id*
19. **show bgp vrf** *vrf-name ip-address*
20. **show ospf vrf** *vrf-name* **database**

DETAILED STEPS

Step 1 **show running-config router bgp** *as-number* **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A
```

Displays the specified VPN routing and forwarding (VRF) content of the currently running configuration.

Step 2 **show running-config routes**

Example:

```
RP/0/RP0/CPU0:router# show running-config routes
```

Displays the Open Shortest Path First (OSPF) routes table in the currently running configuration.

Step 3 **show ospf vrf** *vrf-name* **database**

Example:

```
RP/0/RP0/CPU0:router# show ospf vrf vrf_A database
```

Displays lists of information related to the OSPF database for a specified VRF.

Step 4 **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router# show running-config router bgp 3 vrf vrf_A neighbor 172.168.40.24
```

Displays the Border Gateway Protocol (BGP) VRF neighbor content of the currently running configuration.

Step 5 **show bgp vrf** *vrf-name* **summary**

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A summary
```

Displays the status of the specified BGP VRF connections.

Step 6 **show bgp vrf** *vrf-name* **neighbors** *ip-address*

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A neighbors 172.168.40.24
```

Displays information about BGP VRF connections to the specified neighbors.

Step 7 **show bgp vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A
```

Displays information about a specified BGP VRF.

Step 8 **show route vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0
```

Displays the current routes in the Routing Information Base (RIB) for a specified VRF.

Step 9 **show bgp vpn unicast summary**

Example:

```
RP/0/RP0/CPU0:router# show bgp vpn unicast summary
```

Displays the status of all BGP VPN unicast connections.

Step 10 **show running-config router isis**

Example:

```
RP/0/RP0/CPU0:router# show running-config router isis
```

Displays the Intermediate System-to-Intermediate System (IS-IS) content of the currently running configuration.

Step 11 **show running-config mpls**

Example:

```
RP/0/RP0/CPU0:router# show running-config mpls
```

Displays the MPLS content of the currently running-configuration.

Step 12 **show isis adjacency**

Example:

```
RP/0/RP0/CPU0:router# show isis adjacency
```

Displays IS-IS adjacency information.

Step 13 **show mpls ldp forwarding**

Example:

```
RP/0/RP0/CPU0:router# show mpls ldp forwarding
```

Displays the Label Distribution Protocol (LDP) forwarding state installed in MPLS forwarding.

Step 14 **show bgp vpnv4 unicast** or **show bgp vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0:router# show bgp vpnv4 unicast
```

Displays entries in the BGP routing table for VPNv4 or VPNv6 unicast addresses.

Step 15 **show bgp vrf** *vrf-name* **imported-routes**

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A imported-routes
```

Displays BGP information for routes imported into specified VRF instances.

Step 16 **show route vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show route vrf vrf_A 10.0.0.0
```

Displays the current specified VRF routes in the RIB.

Step 17 **show cef vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1
```

Displays the IPv4 Cisco Express Forwarding (CEF) table for a specified VRF.

Step 18 **show cef vrf** *vrf-name ip-address* **location** *node-id*

Example:

```
RP/0/RP0/CPU0:router# show cef vrf vrf_A 10.0.0.1 location 0/1/cpu0
```

Displays the IPv4 CEF table for a specified VRF and location.

Step 19 **show bgp vrf** *vrf-name ip-address*

Example:

```
RP/0/RP0/CPU0:router# show bgp vrf vrf_A 10.0.0.0
```

Displays entries in the BGP routing table for VRF vrf_A.

Step 20 **show ospf vrf** *vrf-name* **database**

Example:

```
RP/0/RP0/CPU0:router# show ospf vrf vrf_A database
```

Displays lists of information related to the OSPF database for a specified VRF.

Configuring L3VPN over GRE

Perform the following tasks to configure L3VPN over GRE:

Creating a GRE Tunnel between Provider Edge Routers

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

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 path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RP0/CPU0:router(config)# interface tunnel-ip 4000
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

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

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 4 **ipv6 address** *ipv6-prefix/prefix-length*

Example:

```
RP/0/RP0/CPU0:router(config-if)# ipv6 address 100:1:1:1::1/64
```

Specifies an IPv6 network assigned to the interface.

Step 5 **tunnel mode gre ipv4**

Example:

```
RP/0/RP0/CPU0:router(config-if)# tunnel mode gre ipv4
```

Sets the encapsulation mode of the tunnel interface to GRE.

Step 6 **tunnel source** *type path-id*

Example:

```
RP/0/RP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1
```

Specifies the source of the tunnel interface.

Step 7 **tunnel destination** *ip-address*

Example:

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

Defines the tunnel destination.

Step 8 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

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.

Step 9 **interface tunnel-ip** *number*

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

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 10 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

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*
13. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

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

Enters the Global Configuration mode.

Step 2 **mpls ldp**

Example:
```
RP/0/RP0/CPU0:router(config)# mpls ldp
```

Enables MPLS LDP configuration mode.

Step 3 **router-id** { *router-id* }

Example:
```
RP/0/RP0/CPU0:router(config-ldp)# 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 4 **discovery hello holdtime** *seconds*

Example:
```
RP/0/RP0/CPU0:router(config-ldp)# discovery hello holdtime 40
```

Defines the period of time a discovered LDP neighbor is remembered without receipt of an LDP Hello message from the neighbor.

Note We recommend using a stable IP address as the router ID.

Step 5 **discovery hello interval** *seconds*

Example:
```
RP/0/RP0/CPU0:router(config-ldp)# discovery hello holdtime 20
```

Defines the period of time between the sending of consecutive Hello messages.

Step 6 **nsr**

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

Activates BGP NSR.

Step 7 **graceful-restart**

Example:
```
RP/0/RP0/CPU0:router(config-ldp)# graceful-restart
```

Enables graceful restart on the router.

Step 8 **graceful-restart reconnect-timeout** *seconds*

Example:
```
RP/0/RP0/CPU0:router(config-ldp)# graceful-restart recoonect-timeout 180
```

Defines the time for which the neighbor should wait for a reconnection if the LDP session is lost.

Step 9 **graceful-restart forwarding-state-holdtime** *seconds*

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

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

Step 10 **holdtime** *seconds*

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

Configures the hold time for an interface.

Step 11 **neighbor** *ip-address*

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

Defines a neighboring router.

Step 12 **interface tunnel-ip** *number*

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

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 13 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring L3VPN

Perform this task to configure L3VPN.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
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 native vlan** *vlan-id*
10. **router bgp** *as-number*
11. **nsr**
12. **bgp router-id** *ip-address*
13. **address-family {vpnv4 | vpnv6} unicast**
14. **neighbor** *ip-address*
15. **remote-as** *as-number*
16. **update-source** *type interface-path-id*
17. **address-family { vpnv4 | vpnv6 } unicast**
18. **route-policy** *route-policy-name* **in**
19. **route-policy** *route-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. **ebg-multihop** *ttl-value*
28. **address-family { ipv4 | ipv6 } unicast**
29. **route-policy** *route-policy-name* **in**
30. **route-policy** *route-policy-name* **out**
31. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **vrf** *vrf-name*

Example:
```
RP/0/RP0/CPU0:router(config)# vrf vpn1
```

Configures a VRF instance.

Step 3 **address-family { ipv4 | ipv6 } unicast**

Example:
```
RP/0/RP0/CPU0:router(config-vrf)# address-family { ipv4 | ipv6 } unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 4 **import route-target** [*as-number:nn* | *ip-address:nn*]

Example:
```
RP/0/RP0/CPU0:router(config-vrf)# import route-target 2:1
```

Specifies a list of route target (RT) extended communities. Only prefixes that are associated with the specified import route target extended communities are imported into the VRF.

Step 5 **export route-target** [*as-number:nn* | *ip-address:nn*]

Example:
```
RP/0/RP0/CPU0:router(config-vrf)# export route-target 1:1
```

Specifies a list of route target extended communities. Export route target communities are associated with prefixes when they are advertised to remote PEs. The remote PEs import them into VRFs which have import RTs that match these exported route target communities.

Step 6 **interface** *type interface-path-id*

Example:
```
RP/0/RP0/CPU0:router(config)# interface TenGigE0/2/0/0.1
```

Enters interface configuration mode and configures an interface.

Step 7 **vrf** *vrf-name*

Example:
```
RP/0/RP0/CPU0:router(config-if)# vrf vpn1
```

Configures a VRF instance.

Step 8 **ipv4 address** *ipv4-address subnet-mask*

Example:
```
RP/0/RP0/CPU0:router(config-if)# ipv4 address 150.1.1.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 9 **dot1q native vlan** *vlan-id*

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

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

Step 10 **router bgp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 1
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 11 **nsr**

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# nsr
```

Activates BGP NSR.

Step 12 **bgp router-id** *ip-address*

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1
```

Configures the local router with a specified router ID.

Step 13 **address-family {vpnv4 | vpnv6} unicast**

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

Enters address family configuration submode for the specified address family.

Step 14 **neighbor** *ip-address*

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 4.4.4.4
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 15 **remote-as** *as-number*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 1
```

Creates a neighbor and assigns a remote autonomous system number to it..

Step 16 **update-source** *type interface-path-id*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)#update-source Loopback0
```

Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.

Step 17 **address-family { vpnv4 | vpnv6 } unicast**

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

Enters address family configuration submode for the specified address family.

Step 18 **route-policy** *route-policy-name* **in**

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy pass-all in
```

Defines a route policy and enters route policy configuration mode.

Step 19 **route-policy** *route-policy-name* **out**

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy pass-all out
```

Defines a route policy and enters route policy configuration mode.

Step 20 **vrf** *vrf-name*

Example:
```
RP/0/RP0/CPU0:router(config)# vrf vpn1
```

Configures a VRF instance.

Step 21 **rd { as-number:nn | ip-address:nn | auto }**

Example:
```
RP/0/RP0/CPU0:router(config-vrf)#rd 1:1
```

Configures the route distinguisher.

Step 22 **address-family { ipv4 | ipv6 } unicast**

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 23 **redistribute connected** [**metric** *metric-value*] [*route-policy route-policy-name*]

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

Configures the local router with a specified router ID.

Step 24 **redistribute static** [**metric** *metric-value*] [*route-policy route-policy-name*]

Example:
```
RP/0/RP0/CPU0:router(config-vrf-af)#
redistribute static
```

Causes routes from the specified instance to be redistributed into BGP.

Step 25 **neighbor** *ip-address*

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 150.1.1.2
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 26 **remote-as** *as-number*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)#remote-as 7501
```

Creates a neighbor and assigns a remote autonomous system number to it..

Step 27 **ebg-multihop** *ttl-value*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr)
#ebgp-multihop 10
```

Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected.

Step 28 **address-family { ipv4 | ipv6 } unicast**

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

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 29 **route-policy** *route-policy-name* **in**

Example:

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

Configures the local router with a specified router.

Step 30 **route-policy** *route-policy-name* **out**

Example:
```
RP/0/RP0/CPU0:router(config-bgp-nbr-af)#route-policy BGP_pass_all out
```

Defines a route policy and enters route policy configuration mode.

Step 31 Use the **commit** or **end** command.

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

end - Prompts user to take one of these actions:

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

Configuring 6VPE Support

The following tasks are required to configure 6VPE support:

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

DETAILED STEPS

Step 1 **configure**

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

Enters the Global Configuration mode.

Step 2 **vrf** *vrf-name*

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

Configures a VRF instance and enters VRF configuration mode.

Step 3 **address-family ipv6 unicast**

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

Enters VRF address family configuration mode for the IPv6 address family.

Step 4 **import route-target** [*as-number:nn* | *ip-address:nn*]

Example:
```
RP/0/RP0/CPU0:router(config-vrf-af)# import route-target 120:1
```

Configures a VPN routing and forwarding (VRF) import route-target extended community.

Step 5 **export route-target** [*as-number:nn* | *ip-address:nn*]

Example:
```
RP/0/RP0/CPU0:router(config-vrf-af)# export route-target 120:2
```

Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 6 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring BGP Route Distinguisher and Core-facing Sessions

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

Note Before you perform this task, you must first configure a VRF and map the VRF to an interface. For more information, see Implementing MPLS VPNs over IP Tunnels on Cisco IOS XR Software.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **address-family ipv6 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. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

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

Enters the Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:
```
RP/0/RP0/CPU0:router(config)# router bgp 100
RP/0/RP0/CPU0:router(config-bgp)#
```

Enters router BGP configuration mode.

Step 3 **address-family ipv6 unicast**

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
RP/0/RP0/CPU0:router(config-bgp-af)
```

Enters address family configuration mode for the VPNv6 address family.

Step 4 **vrf** *vrf-name*

Example:
```
RP/0/RP0/CPU0:router(config-bgp)# vrf red
```

Configures a VPN VRF instance and enters VRF configuration mode.

Step 5 **rd** { *as-number:nn* | *ip-address:nn* **auto** }

Example:
```
RP/0/RP0/CPU0:router(config-bgp-vrf)# router bgp 100
```

Configures a route distinguisher.

Step 6 **address-family ipv6 unicast**

Example:
```
RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast
```

Enters IPv6 address family configuration mode.

Step 7 **exit**

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

Exits the current configuration mode.

Step 8 **neighbor** *ip-address* **remote-as** *as-number*

Example:
```
RP/0/RP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24 remote-as 2002f
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 9 **address-family ipv6 unicast**

Example:
```
RP/0/RP0/CPU0:router(config-bgp-vrf)# address-family ipv6 unicast
```

Enters IPv6 address family configuration mode.

Step 10 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuring a PE-CE Protocol

Perform this task to configure a PE-CE protocol for 6VPE.

Note eBGP, iBGP and eiBGP load-balancing configuration options are also supported for 6VPE.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv6 unicast**
5. **exit**
6. **exit**
7. **neighbor** *ip-address*
8. **remote-as** *as-number*
9. **address-family vpnv6 unicast**
10. **route-policy** *route-policy-name* **in**
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

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

Enters the Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config)# router bgp 120
RP/0/RP0/CPU0:router(config-bgp)
```

Enters router BGP configuration mode.

Step 3 **vrf** *vrf-name*

Example:

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

Configures a VPN VRF instance and enters VRF configuration mode.

Step 4 **address-family ipv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-vrf) address-family ipv6 unicast
RP/0/RP0/CPU0:router(config-bgp-vrf-af)
```

Enters IPv6 address family configuration mode.

Step 5 **exit**

Example:

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

Exits the current configuration mode.

Step 6 **exit**

Example:

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

Exits the current configuration mode.

Step 7 **neighbor** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-bgp)# neighbor 10,10.10,10
```

Creates a neighbor and assigns it a remote autonomous system number of 2002.

Step 8 **remote-as** *as-number*

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# remote-as 1000
```

Creates a BGP neighbor and begin the exchange of routing information.

Step 9 **address-family vpnv6 unicast**

Example:

```
RP/0/RP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/RP0/CPU0:router(config-bgp-nbr-af)
```

Enters address family configuration mode for the VPNv6 address family.

Step 10 **route-policy** *route-policy-name* **in**

Example:

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

Applies a routing policy to updates advertised to or received from a BGP neighbor.

Step 11 Use the **commit** or **end** command.
commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

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

Configuration Examples for Implementing MPLS Layer 3 VPNs

The following section provides sample configurations for MPLS L3VPN features:

Configuring an MPLS VPN Using BGP: Example

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

```
address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!
interface Loopback0
  ipv4 address 10.0.0.1 255.255.255.255
!
interface TenGigE 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!
interface TenGigE 0/1/0/1
  ipv4 address 10.0.0.1 255.0.0.0
!
router ospf 100
  area 100
    interface loopback0
    interface TenGigE 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 looback0
  interface TenGigE 0/1/0/1
!
```

Configuring the Routing Information Protocol on the PE Router: Example

The following example shows the configuration for the RIP on the PE router:

```
vrf vpn1
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!

interface TenGigE 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!

router rip
 vrf vpn1
  interface TenGigE 0/1/0/0
  !
  timers basic 30 90 90 120
  redistribute bgp 100
  default-metric 3
  route-policy pass-all in
 !
```

Configuring the PE Router Using EIGRP: Example

The following example shows the configuration for the Enhanced Interior Gateway Routing Protocol (EIGRP) on the PE router:

```
Router eigrp 10
 vrf VRF1
  address-family ipv4
   router-id 10.1.1.2
   default-metric 100000 2000 255 1 1500
   as 62
   redistribute bgp 2000
   interface Loopback0
   !
   interface TenGigE 0/6/0/0
```

Configuration Examples for MPLS VPN CSC

Configuration examples for the MPLS VPN CSC include:

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

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

In this example, a CSC-PE router peers with a PE router, 10.1.0.2, in its own AS. It also has a labeled unicast peering with a CSC-CE router, 10.0.0.1.

```
config
 router bgp 2
  address-family vpnv4 unicast
  neighbor 10.1.0.2
      remote-as 2
     update-source loopback0
     address-family vpnv4 unicast
    vrf customer-carrier
     rd 1:100
     address-family ipv4 unicast
      allocate-label all
      redistribute static
    neighbor 10.0.0.1
     remote-as 1
     address-family ipv4 labeled-unicast
      route-policy pass-all in
      route-policy pass-all out
      as-override
end
```

Configuring a CSC-CE: Example

The following example shows how to configure a CSC-CE router. In this example, the CSC-CE router peers CSC-PE router 10.0.0.2 in AS 2.

```
config
 router bgp 1
  address-family ipv4 unicast
     redistribute ospf 200
     allocate-label all
    neighbor 10.0.0.2
     remote-as 2
     address-family ipv4 labeled-unicast
     route-policy pass-all in
     route-policy pass-all out
end
```

Configuring a Static Route to a Peer: Example

The following example shows how to configure a static route to an Inter-AS or CSC-CE peer:

```
config
```

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

Configuring a Static Route to a Peer: Example

This example shows how to configure a static route to an Inter-AS or CSC-CE peer:

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

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:

```
RP/0/RSP0/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::1/64
 tunnel mode gre ipv4
 tunnel source TenGigE0/2/0/1
 tunnel destination 145.12.5.2
!
RP/0/RSP0/CPU0:PE2#sh run int tunnel-ip 1
interface tunnel-ip1
 ipv4 address 100.1.1.2 255.255.255.0
 ipv6 address 100:1:1:1::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.

```
RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:PE1#sh 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/RSP0/CPU0:PE1#sh run vrf vpn1
vrf vpn1
 address-family ipv4 unicast
  import route-target
   2:1
  !
  export route-target
   1:1
  !
RP/0/RSP0/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
 encapsulation dot1q 1
!

RP/0/RSP0/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/RSP0/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 adverstised and verify that mpls counters increase in tunnel
interface accounting

RP/0/RSP0/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

The following example shows a standard configuration of an IPv6 address family under VRF:

```
configure
 vrf red
  address-family ipv6 unicast
   import route-target
    500:1
   !
   export route-target
    500:1
   !
  !
```

Configuring BGP for the Address Family VPNv6: Example

The following example shows the configuration for the address family VPNv6 under the PE peer:

```
configure
 router bgp 3
  address-family vpnv6 unicast
  !
   neighbor 192.168.254.3
   remote-as 3
   update-source Loopback0
   address-family ipv4 unicast
   !
   address-family vpnv44 unicast
   !
   address-family ipv6 labeled-unicast
   !
   address-family vpnv6 unicast
   !
  !
```

Configuring the Address Family IPv6 for the VRF Configuration Under BGP: Example

The following example shows the configuration for the address family IPv6 for the VRF configuration under BGP:

```
!
 vrf red
  address-family ipv6 unicast
  redistribute connected
  !
```

Configuring a PE-CE Protocol: Example

The following example shows the eBGP configuration of a PE-CE protocol:

```
  !
  neighbor 2001:db80:cafe:1::2
   remote-as 100
   address-family ipv6 unicast
   route-policy pass in
   route-policy pass out
```

Configuring an Entire 6VPE Configuration: Example

Two VPNs, which are named red and blue, are created across router2 and router4. The VRF red is for the user running IPv6 addressing in the network. The VRF blue is for the user running IPv4 addressing. 6VPE is implemented to carry the VPNv6 prefixes across to the other PE.

The following example shows the entire 6VPE configuration that includes the interface and VRF configurations of both PE routers across the route reflectors:

```
router2 (PE router)
interface GigabitEthernet0/0/1/3.1
 vrf red
 ipv4 address 192.3.1.1 255.255.255.0
 ipv6 address 2001:db80:cafe:1::1/64
 dot1q vlan 2
!

show run interface gigabitEthernet 0/0/1/3.2
interface GigabitEthernet0/0/1/3.2
 vrf blue
```

```
 ipv4 address 192.3.2.1 255.255.255.0
 dot1q vlan 3
!


vrf red
 address-family ipv4 unicast
 import route-target
  500:1
 !
 export route-target
  500:1
 !
!
 address-family ipv6 unicast
 import route-target
  500:1
 !
 export route-target
  500:1
  !
 !
!
vrf blue
 address-family ipv4 unicast
  import route-target
   600:1
  !
  export route-target
   600:1
  !
 !


router bgp 3
 address-family ipv4 unicast
  network 3.3.3.3/32
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
  network 2001:db82:cafe:1::/64
  allocate-label all
 !
address-family vpnv6 unicast
 !
 neighbor 192.168.253.4
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
```

```
  !
 !
 vrf red
  rd 500:1
  address-family ipv4 unicast
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
  neighbor 2001:db80:cafe:1::2
   remote-as 100
   address-family ipv6 unicast
    route-policy pass in
    route-policy pass out
   !
  !
 !
 vrf blue
  rd 600:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
!


router3 (RR)

router bgp 3
 bgp router-id 192.168.253.4
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
 !
 address-family vpnv6 unicast
 !
 neighbor-group all
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
   route-reflector-client
  !
  address-family vpnv4 unicast
   route-reflector-client
  !
  address-family ipv6 labeled-unicast
   route-reflector-client
  !
  address-family vpnv6 unicast
   route-reflector-client
  !
 !
 neighbor 192.168.253.1
  use neighbor-group all
 !
 neighbor 192.168.253.2
  use neighbor-group all
 !
 neighbor 192.168.253.3
  use neighbor-group all
 !
 neighbor 192.168.253.5
  use neighbor-group all
 !
 neighbor 192.168.253.6
  use neighbor-group all
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
```

```
  address-family ipv4 unicast
  !
 !
!

router4(PE router)

vrf red
 address-family ipv4 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
 address-family ipv6 unicast
  import route-target
   500:1
  !
  export route-target
   500:1
  !
 !
!
vrf blue
 address-family ipv4 unicast
  import route-target
   600:1
  !
  export route-target
   600:1
  !
 !
!


router bgp 3
 address-family ipv4 unicast
 !
 address-family vpnv4 unicast
 !
 address-family ipv6 unicast
  network 2001:db84:beef:1::/64
  allocate-label all
 !
 address-family vpnv6 unicast
 !
 neighbor 192.168.253.4
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
  address-family vpnv6 unicast
  !
 !
 neighbor 192.168.254.3
  remote-as 3
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv4 unicast
  !
  address-family ipv6 labeled-unicast
  !
 !
 vrf red
  rd 500:1
  address-family ipv4 unicast
```

```
   redistribute connected
  !
  address-family ipv6 unicast
   redistribute connected
  !
 !
 vrf blue
  rd 600:1
  address-family ipv4 unicast
   redistribute connected
  !
 !
!
```

The following example displays the sample output for the entire 6VPE configuration:

```
show route vrf red ipv6

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

C    2001:db80:beef:1::/64 is directly connected,
      19:09:50, GigabitEthernet0/0/1/3.1
L    2001:db80:beef:1::1/128 is directly connected,
      19:09:50, GigabitEthernet0/0/1/3.1
B    2001:db80:cafe:1::/64
      [200/0] via ::ffff:192.168.253.3 (nexthop in vrf default), 07:03:40


show route vrf red ipv6

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local

Gateway of last resort is not set

B    2001:db80:beef:1::/64
      [200/0] via ::ffff:192.168.253.6 (nexthop in vrf default), 07:04:14
C    2001:db80:cafe:1::/64 is directly connected,
      08:28:12, GigabitEthernet0/0/1/3.1
L    2001:db80:cafe:1::1/128 is directly connected,
      08:28:12, GigabitEthernet0/0/1/3.1
```