Multiprotocol Label Switching (MPLS) Configuration Guide, Cisco IOS XE Gibraltar 16.11.x (Catalyst 9500 Switches)

First Published: 2019-03-29

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

CONTENTS

CHAPTER 1

Configuring Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching

This module describes Multiprotocol Label Switching and how to configure it on Cisco switches.

Restrictions for Multiprotocol Label Switching

- Multiprotocol Label Switching (MPLS) fragmentation is not supported.
- MPLS maximum transmission unit (MTU) is not supported.

Information about Multiprotocol Label Switching

Multiprotocol label switching (MPLS) combines the performance and capabilities of Layer 2 (data link layer) switching with the proven scalability of Layer 3 (network layer) routing. MPLS enables you to meet the challenges of explosive growth in network utilization while providing the opportunity to differentiate services without sacrificing the existing network infrastructure. The MPLS architecture is flexible and can be employed in any combination of Layer 2 technologies. MPLS support is offered for all Layer 3 protocols, and scaling is possible well beyond that typically offered in today’s networks.

Functional Description of Multiprotocol Label Switching

Label switching is a high-performance packet forwarding technology that integrates the performance and traffic management capabilities of data link layer (Layer 2) switching with the scalability, flexibility, and performance of network layer (Layer 3) routing.

Label Switching Functions

In conventional Layer 3 forwarding mechanisms, as a packet traverses the network, each switch extracts all the information relevant to forwarding the packet from the Layer 3 header. This information is then used as an index for a routing table lookup to determine the next hop for the packet.

In the most common case, the only relevant field in the header is the destination address field, but in some cases, other header fields might also be relevant. As a result, the header analysis must be done independently at each switch through which the packet passes. In addition, a complicated table lookup must also be done at each switch.

In label switching, the analysis of the Layer 3 header is done only once. The Layer 3 header is then mapped into a fixed length, unstructured value called a *label* .

Many different headers can map to the same label, as long as those headers always result in the same choice of next hop. In effect, a label represents a *forwarding equivalence class* --that is, a set of packets which, however different they may be, are indistinguishable by the forwarding function.

The initial choice of a label need not be based exclusively on the contents of the Layer 3 packet header; for example, forwarding decisions at subsequent hops can also be based on routing policy.

Once a label is assigned, a short label header is added at the front of the Layer 3 packet. This header is carried across the network as part of the packet. At subsequent hops through each MPLS switch in the network, labels are swapped and forwarding decisions are made by means of MPLS forwarding table lookup for the label carried in the packet header. Hence, the packet header does not need to be reevaluated during packet transit through the network. Because the label is of fixed length and unstructured, the MPLS forwarding table lookup process is both straightforward and fast.

Distribution of Label Bindings

Each label switching router (LSR) in the network makes an independent, local decision as to which label value to use to represent a forwarding equivalence class. This association is known as a label binding. Each LSR informs its neighbors of the label bindings it has made. This awareness of label bindings by neighboring switches is facilitated by the following protocols:

- Label Distribution Protocol (LDP)--enables peer LSRs in an MPLS network to exchange label binding information for supporting hop-by-hop forwarding in an MPLS network
- Border Gateway Protocol (BGP)--Used to support MPLS virtual private networks (VPNs)

When a labeled packet is being sent from LSR A to the neighboring LSR B, the label value carried by the IP packet is the label value that LSR B assigned to represent the forwarding equivalence class of the packet. Thus, the label value changes as the IP packet traverses the network.

For more information about LDP configuration, see the see MPLS: LDP Configuration Guide at http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mpls/config_library/xe-3s/mp-xe-3s-library.html

Note As the scale of label entries is limited in, especially with ECMP, it is recommended to enable LDP label filtering. LDP labels shall be allocated only for well known prefixes like loopback interfaces of routers and any prefix that needs to be reachable in the global routing table.

MPLS Layer 3 VPN

A Multiprotocol Label Switching (MPLS) 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.

Before configuring MPLS Layer 3 VPNs, you should have MPLS, Label Distribution Protocol (LDP), and Cisco Express Forwarding (CEF) installed in your network. All routers in the core, including the PE routers, must be able to support CEF and MPLS forwarding.

Classifying and Marking MPLS QoS EXP

The QoS EXP Matching feature allows you to classify and mark network traffic by modifying the Multiprotocol Label Switching (MPLS) experimental bits (EXP) field in IP packets.

The QoS EXP Matching feature allows you to organize network traffic by setting values for the MPLS EXP field in MPLS packets. By choosing different values for the MPLS EXP field, you can mark packets so that packets have the priority that they require during periods of congestion. Setting the MPLS EXP value allows you to:

- **Classify traffic:** The classification process selects the traffic to be marked. Classification accomplishes this by partitioning traffic into multiple priority levels, or classes of service. Traffic classification is the primary component of class-based QoS provisioning.
- **Police and mark traffic**: Policing causes traffic that exceeds the configured rate to be discarded or marked to a different drop level. Marking traffic is a way to identify packet flows to differentiate them. Packet marking allows you to partition your network into multiple priority levels or classes of service.

Restrictions

Following is the list of restrictions for classifying and marking MPLS QoS EXP:

- Only Uniform mode and Pipe mode are supported; Short-pipe mode is not supported.
- Support range of QoS-group values range between 0 and 30. (Total 31 QoS-groups).
- EXP marking using QoS policy is supported only on the outer label; inner EXP marking is not supported.

How to Configure Multiprotocol Label Switching

This section explains how to perform the basic configuration required to prepare a switch for MPLS switching and forwarding.

Configuring a Switch for MPLS Switching

MPLS switching on Cisco switches requires that Cisco Express Forwarding be enabled.

> **Note** **ip unnumbered** command is not supported in MPLS configuration.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip cef distributed** **Example:** `Device(config)# ip cef distributed`	Enables Cisco Express Forwarding on the switch.
Step 4	**mpls label range** *minimum-value maximum-value* **Example:** `Device(config)# mpls label range 16 4096`	Configure the range of local labels available for use with MPLS applications on packet interfaces.
Step 5	**mpls label protocol ldp** **Example:** `Device(config)# mpls label protocol ldp`	Specifies the label distribution protocol for the platform.

Configuring a Switch for MPLS Forwarding

MPLS forwarding on Cisco switches requires that forwarding of IPv4 packets be enabled.

> **Note** **ip unnumbered** command is not supported in MPLS configuration.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot/subslot /port* **Example:** `Device(config)# interface gigabitethernet 1/0/0`	Specifies the Gigabit Ethernet interface and enters interface configuration mode. For Switch Virtual Interface (SVI), the example is `Device(config)# interface vlan 1000`
Step 4	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS forwarding of IPv4 packets along routed physical interfaces (Gigabit Ethernet), Switch Virtual Interface (SVI), or port channels.
Step 5	**mpls label protocol ldp** **Example:** `Device(config-if)# mpls label protocol ldp`	Specifies the label distribution protocol for an interface. **Note** MPLS LDP cannot be enabled on a Virtual Routing and Forwarding (VRF) interface.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Verifying Multiprotocol Label Switching Configuration

This section explains how to verify successful configuration of MPLS switching and forwarding.

Verifying Configuration of MPLS Switching

To verify that Cisco Express Forwarding has been configured properly, issue the **show ip cef summary** command, which generates output similar to that shown below:

Procedure

show ip cef summary

Example:

```
Device# show ip cef summary

IPv4 CEF is enabled for distributed and running
VRF Default
 150 prefixes (149/1 fwd/non-fwd)
 Table id 0x0
 Database epoch:        4 (150 entries at this epoch)
Device#
```

Verifying Configuration of MPLS Forwarding

To verify that MPLS forwarding has been configured properly, issue the **show mpls interfaces detail** command, which generates output similar to that shown below:

> **Note** The MPLS MTU value is equivalent to the IP MTU value of the port or switch by default. MTU configuration for MPLS is not supported.

Procedure

Step 1 **show mpls interfaces detail**

Example:

```
For physical (Gigabit Ethernet) interface:
Device# show mpls interfaces detail interface GigabitEthernet 1/0/0

        Type Unknown
        IP labeling enabled
        LSP Tunnel labeling not enabled
        IP FRR labeling not enabled
        BGP labeling not enabled
        MPLS not operational
        MTU = 1500

For Switch Virtual Interface (SVI):
Device# show mpls interfaces detail interface Vlan1000

        Type Unknown
        IP labeling enabled (ldp) :
          Interface config
        LSP Tunnel labeling not enabled
        IP FRR labeling not enabled
        BGP labeling not enabled
        MPLS operational
        MTU = 1500
```

Step 2 **show running-config interface**

Example:

```
For physical (Gigabit Ethernet) interface:
Device# show running-config interface interface GigabitEthernet 1/0/0

Building configuration...


Current configuration : 307 bytes
!
interface TenGigabitEthernet1/0/0
no switchport
ip address xx.xx.x.x xxx.xxx.xxx.x
mpls ip
mpls label protocol ldp
end

For Switch Virtual Interface (SVI):
Device# show running-config interface interface Vlan1000

Building configuration...


Current configuration : 187 bytes
!
interface Vlan1000
ip address xx.xx.x.x xxx.xxx.xxx.x
mpls ip
mpls label protocol ldp
end
```

Step 3 **show mpls forwarding**

Example:

```
For physical (Gigabit Ethernet) interface:
Device# show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
500        No Label   l2ckt(3)         0             Gi3/0/22   point2point
501        No Label   l2ckt(1)         12310411816789 none       point2point
502        No Label   l2ckt(2)         0             none       point2point
503        566        15.15.15.15/32   0             Po5        192.1.1.2
504        530        7.7.7.7/32       538728528     Po5        192.1.1.2
505        573        6.6.6.10/32      0             Po5        192.1.1.2
506        606        6.6.6.6/32       0             Po5        192.1.1.2
507        explicit-n 1.1.1.1/32       0             Po5        192.1.1.2
556        543        19.10.1.0/24     0             Po5        192.1.1.2
567        568        20.1.1.0/24      0             Po5        192.1.1.2
568        574        21.1.1.0/24      0             Po5        192.1.1.2
574        No Label   213.1.1.0/24[V]  0             aggregate/vpn113
575        No Label   213.1.2.0/24[V]  0             aggregate/vpn114
576        No Label   213.1.3.0/24[V]  0             aggregate/vpn115
577        No Label   213:1:1::/64     0             aggregate
594        502        103.1.1.0/24     0             Po5        192.1.1.2
595        509        31.1.1.0/24      0             Po5        192.1.1.2
596        539        15.15.1.0/24     0             Po5        192.1.1.2
597        550        14.14.1.0/24     0             Po5        192.1.1.2
633        614        2.2.2.0/24       0             Po5        192.1.1.2
634        577        90.90.90.90/32   873684        Po5        192.1.1.2
635        608        154.1.1.0/24     0             Po5        192.1.1.2
```

```
636       609       153.1.1.0/24    0            Po5       192.1.1.2
Device# end
```

Additional References for Multiprotocol Label Switching

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the Multiprotocol Label Switching (MPLS) Commands section of the *Command Reference (Catalyst 9500 Series Switches)*

Feature History for Multiprotocol Label Switching

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Multiprotocol Label Switching	Multiprotocol Label Switching combines the performance and capabilities of Layer 2 (data link layer) switching with the proven scalability of Layer 3 (network layer) routing.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 2

Configuring eBGP and iBGP Multipath

BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

The BGP Multipath Load Sharing for eBGP and iBGP feature allows you to configure multipath load balancing with both external BGP (eBGP) and internal BGP (iBGP) paths in Border Gateway Protocol (BGP) networks that are configured to use Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs). This feature provides improved load balancing deployment and service offering capabilities and is useful for multi-homed autonomous systems and Provider Edge (PE) routers that import both eBGP and iBGP paths from multihomed and stub networks.

Prerequisites for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Cisco Express Forwarding (CEF) or distributed CEF (dCEF) must be enabled on all participating devices.

Restrictions for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Address Family Support

This feature is configured on a per VPN routing and forwarding instance (VRF) basis. This feature can be configured under both IPv4 and IPv6 VRF address families.

Memory Consumption Restriction

Each BGP multipath routing table entry will use additional memory. We recommend that you do not use this feature on a device with a low amount of available memory and especially if the device carries full Internet routing tables.

Number of Paths Limitation

The number of paths supported are limited to 2 BGP multipaths. This could either be 2 iBGP multipaths or 1 iBGP multipath and 1 eBGP multipath.

Unsupported Commands

ip unnumbered command is not supported in MPLS configuration.

Information About BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Multipath Load Sharing Between eBGP and iBGP

A BGP routing process will install a single path as the best path in the routing information base (RIB) by default. The **maximum-paths** command allows you to configure BGP to install multiple paths in the RIB for multipath load sharing. BGP uses the best path algorithm to select a single multipath as the best path and advertise the best path to BGP peers.

Note The number of paths of multipaths that can be configured is documented on the maximum-paths command reference page.

Load balancing over the multipaths is performed by CEF. CEF load balancing is configured on a per-packet round robin or on a per session (source and destination pair) basis. For information about CEF, refer to Cisco IOS IP Switching Configuration Guide documentation: IP Switching Cisco Express Forwarding Configuration Guide. The BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature is enabled under the IPv4 VRF address family and IPv6 VRF address family configuration modes. When enabled, this feature can perform load balancing on eBGP and/or iBGP paths that are imported into the VRF. The number of multipaths is configured on a per VRF basis. Separate VRF multipath configurations are isolated by unique route distinguisher.

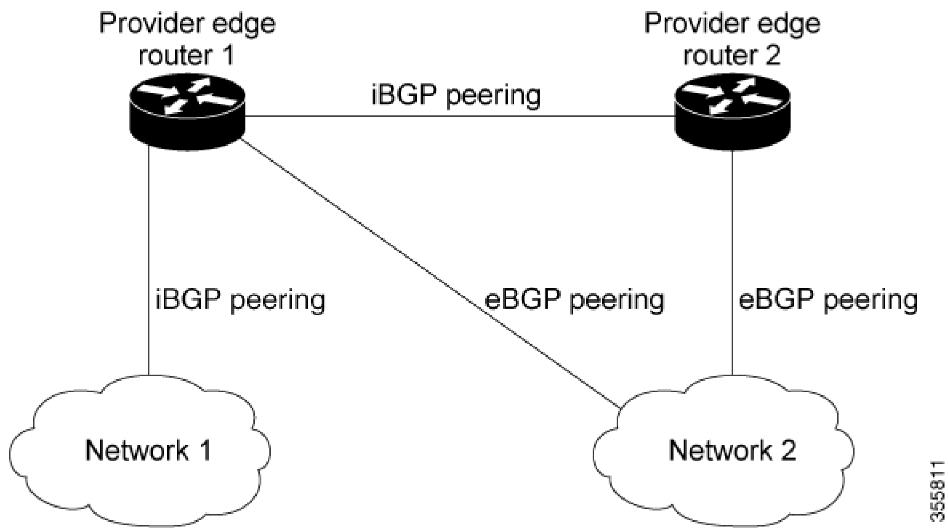
Note The BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature operates within the parameters of configured outbound routing policy.

eBGP and iBGP Multipath Load Sharing in a BGP MPLS Network

The following figure shows a service provider BGP MPLS network that connects two remote networks to PE router 1 and PE router 2. PE router 1 and PE router 2 are both configured for VPNv4 unicast iBGP peering.

Network 2 is a multihomed network that is connected to PE router 1 and PE router 2. Network 2 also has extranet VPN services configured with Network 1. Both Network 1 and Network 2 are configured for eBGP peering with the PE routers.

Figure 1: Service Provider BGP MPLS Network



PE router 1 can be configured with the BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature so that both iBGP and eBGP paths can be selected as multipaths and imported into the VRF . The multipaths will be used by CEF to perform load balancing. IP traffic that is sent from Network 1 to Network 2, PE router 1 will Load Share with eBGP paths as IP traffic & iBGP path will be sent as MPLS traffic.

Note

- eBGP session between local CE & local PE is not supported.
- eBGP session from a local PE to a remote CE is supported.
- eiBGP Multipath is supported in per prefix label allocation mode only. It is not supported in other label allocation modes.

Benefits of Multipath Load Sharing for Both eBGP and iBGP

The BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS VPN feature allows multihomed autonomous systems and PE routers to be configured to distribute traffic across both eBGP and iBGP paths.

How to Configure BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

This section contains the following procedures:

Configuring Multipath Load Sharing for Both eBGP an iBGP

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure** { **terminal** | **memory** | **network** } **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp as-number** **Example:** `Device(config)# router bgp 40000`	Enters router configuration mode to create or configure a BGP routing process.
Step 4	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name* } **Example:** `Device(config-router)# neighbor group192`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected.
Step 5	**address-family ipv4 vrf***vrf-name* **Example:** `Device(config-router)# address-family ipv4 vrf RED`	Places the router in address family configuration mode. • Separate VRF multipath configurations are isolated by unique route distinguisher.
Step 6	**address-family ipv6 vrf***vrf-name* **Example:** `Device(config-router)# address-family ipv6 vrf RED`	Places the router in address family configuration mode. • Separate VRF multipath configurations are isolated by unique route distinguisher.
Step 7	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name* } **update-source** *interface-type interface-name* **Example:** `Device(config-router)# neighbor FE80::1234:BFF:FE0E:A471 update-source Gigabitethernet 1/0/0`	Specifies the link-local address over which the peering is to occur.
Step 8	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name* } **activate** **Example:** `(config-router)# neighbor group192 activate`	Activates the neighbor or listen range peer group for the configured address family.

	Command or Action	Purpose
Step 9	**maximum-paths eibgp** [*import-number*] **Example:** `(config-router-af)# maximum-paths eibgp 2`	Configures the number of parallel iBGP and eBGP routes that can be installed into a routing table.

Verifying Multipath Load Sharing for Both eBGP an iBGP

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Displays information about the TCP and BGP connections to neighbors.
Step 3	**show ip bgp vpnv4 vrf***vrf name* **Example:** `Device# show ip bgp vpnv4 vrf RED`	Displays VPN address information from the BGP table. This command is used to verify that the VRF has been received by BGP.
Step 4	**show ip route vrf***vrf-name* **Example:** `Device# show ip route vrf RED`	Displays the IP routing table associated with a VRF instance. The show ip route vrf command is used to verify that the VRF is in the routing table.

Configuration Examples for the BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN Feature

The following examples show how to configure and verify this feature:

eBGP and iBGP Multipath Load Sharing Configuration Example

This following configuration example configures a router in IPv4 address-family mode to select two BGP routes (eBGP or iBGP) as multipaths:

```
Device(config)# router bgp 40000
Device(config-router)# address-family ipv4 vrf RED
Device(config-router-af)# maximum-paths eibgp 2
Device(config-router-af)# end
```

This following configuration example configures a router in IPv6 address-family mode to select two BGP routes (eBGP or iBGP) as multipaths:

```
Device(config)#router bgp 40000
Device(config-router)# address-family ipv6 vrf RED
Device(config-router-af)# maximum-paths eibgp 2
Device(config-router-af)# end
```

Feature Information for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 1: Feature Information for BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN

Feature Name	Releases	Feature Information
BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN	Cisco IOS XE Everest 16.6.1	The BGP Multipath Load Sharing for eBGP and iBGP feature allows you to configure multipath load balancing with both external BGP (eBGP) and internal BGP (iBGP) paths in Border Gateway Protocol (BGP) networks that are configured to use Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs). This feature provides improved load balancing deployment and service offering capabilities and is useful for multi-homed autonomous systems and Provider Edge (PE) routers that import both eBGP and iBGP paths from multihomed and stub networks.

Feature Name	Releases	Feature Information
BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN	Cisco IOS XE Fuji 16.9.1	The BGP Multipath Load Sharing for eBGP and iBGP feature allows you to configure multipath load balancing with both external BGP (eBGP) and internal BGP (iBGP) paths in Border Gateway Protocol (BGP) networks that are configured to use Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs). This feature provides improved load balancing deployment and service offering capabilities and is useful for multi-homed autonomous systems and Provider Edge (PE) routers that import both eBGP and iBGP paths from multihomed and stub networks. This feature was implemented on Cisco Catalyst 9500 Series Switches - High Performance.

CHAPTER 3

Configuring EIGRP MPLS VPN PE-CE Site of Origin

EIGRP MPLS VPN PE-CE Site of Origin

The EIGRP MPLS VPN PE-CE Site of Origin feature introduces the capability to filter Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) traffic on a per-site basis for Enhanced Interior Gateway Routing Protocol (EIGRP) networks. Site of Origin (SoO) filtering is configured at the interface level and is used to manage MPLS VPN traffic and to prevent transient routing loops from occurring in complex and mixed network topologies. This feature is designed to support the MPLS VPN Support for EIGRP Between Provider Edge (PE) and Customer Edge (CE) feature. Support for backdoor links is provided by this feature when installed on PE routers that support EIGRP MPLS VPNs.

Prerequisites for EIGRP MPLS VPN PE-CE Site of Origin

This document assumes that Border GatewayProtocol (BGP) is configured in the network core (or the service provider backbone). The following tasks will also need to be completed before you can configure this feature:

- This feature was introduced to support the MPLS VPN Support for EIGRP Between Provider Edge and Customer Edge feature and should be configured after the EIGRP MPLS VPN is created.
- All PE routers that are configured to support the EIGRP MPLS VPN must run Cisco IOS XE Gibraltar 16.11.1 or a later release, which provides support for the SoO extended community.

Restrictions for EIGRP MPLS VPN PE-CE Site of Origin

- If a VPN site is partitioned and the SoO extended community attribute is configured on a backdoor router interface, the backdoor link cannot be used as an alternate path to reach prefixes originated in other partitions of the same site

- A uniqueSoO value must be configured for each individual VPN site. The same value must be configured on all provider edge and customer edge interfaces (if SoO is configured on the CE routers) that support the same VPN site.
- **ip unnumbered** command is not supported in MPLS configuration.

Information About EIGRP MPLS VPN PE-CE Site of Origin

The following section describes information about EIGRP MPLS VPN PE-CE Site of Origin.

EIGRP MPLS VPN PE-CE Site of Origin Support Overview

The EIGRP MPLS VPN PE-CE Site of Origin feature introduces SoO support for EIGRP-to-BGP and BGP-to-EIGRP redistribution. The SoO extended community is a BGP extended community attribute that is used to identify routes that have originated from a site so that the readvertisement of that prefix back to the source site can be prevented. The SoO extended community uniquely identifies the site from which a PE router has learned a route. SoO support provides the capability to filter MPLS VPN traffic on a per-EIGRP-site basis. SoO filtering is configured at the interface level and is used to manage MPLS VPN traffic and to prevent routing loops from occurring in complex and mixed network topologies, such as EIGRP VPN sites that contain both VPN and backdoor links.

The configuration of the SoO extended community allows MPLS VPN traffic to be filtered on a per-site basis. The SoO extended community is configured in an inbound BGP route map on the PE router and is applied to the interface. The SoO extended community can be applied to all exit points at the customer site for more specific filtering but must be configured on all interfaces of PE routers that provide VPN services to CE routers.

Site of Origin Support for Backdoor Links

The EIGRP MPLS VPN PE-CE Site of Origin (SoO) feature introduces support for backdoor links. A backdoor link or a route is a connection that is configured outside of the VPN between a remote and main site; for example, a WAN leased line that connects a remote site to the corporate network. Backdoor links are typically used as back up routes between EIGRP sites if the VPN link is down or not available. A metric is set on the backdoor link so that the route though the backdoor router is not selected unless there is a VPN link failure.

The SoO extended community is defined on the interface of the backdoor router. It identifies the local site ID, which should match the value that is used on the PE routers that support the same site. When the backdoor router receives an EIGRP update (or reply) from a neighbor across the backdoor link, the router checks the update for an SoO value. If the SoO value in the EIGRP update matches the SoO value on the local backdoor interface, the route is rejected and not added to the EIGRP topology table. This scenario typically occurs when the route with the local SoO valued in the received EIGRP update was learned by the other VPN site and then advertised through the backdoor link by the backdoor router in the other VPN site. SoO filtering on the backdoor link prevents transient routing loops from occurring by filtering out EIGRP updates that contain routes that carry the local site ID.

If this feature is enabled on the PE routers and the backdoor routers in the customer sites, and SoO values are defined on both the PE and backdoor routers, both the PE and backdoor routers will support convergence between the VPN sites. The other routers in the customer sites need only propagate the SoO values carried by the routes, as the routes are forwarded to neighbors. These routers do not otherwise affect or support convergence beyond normal Diffusing Update Algorithm (DUAL) computations.

Router Interoperation with the Site of Origin Extended Community

The configuration of an SoO extended community allows routers that support EIGRP MPLS VPN PE-CE Site of Origin feature to identify the site from which each route originated. When this feature is enabled, the EIGRP routing process on the PE or CE router checks each received route for the SoO extended community and filters based on the following conditions:

- A received route from BGP or a CE router contains an SoO value that matches the SoO value on the receiving interface : If a route is received with an associated SoO value that matches the SoO value that is configured on the receiving interface, the route is filtered because it was learned from another PE router or from a backdoor link. This behavior is designed to prevent routing loops.
- A received route from a CE router is configured with an SoO value that does not match: If a route is received with an associated SoO value that does not match the SoO value that is configured on the receiving interface, the route is added to the EIGRP topology table so that it can be redistributed into BGP. If the route is already installed to the EIGRP topology table but is associated with a different SoO value, the SoO value from the topology table will be used when the route is redistributed into BGP.
- A received route from a CE router does not contain an SoO value: If a route is received without a SoO value, the route is accepted into the EIGRP topology table, and the SoO value from the interface that is used to reach the next hop CE router is appended to the route before it is redistributed into BGP.

When BGP and EIGRP peers that support the SoO extended community receive these routes, they will also receive the associated SoO values and pass them to other BGP and EIGRP peers that support the SoO extended community. This filtering is designed to prevent transient routes from being relearned from the originating site, which prevents transient routing loops from occurring.

Redistribution of BGP VPN Routes That Carry the Site of Origin into EIGRP

When an EIGRP routing process on a PE router redistributes BGP VPN routes into an EIGRP topology table, EIGRP extracts the SoO value (if one is present) from the appended BGP extended community attributes and appends the SoO value to the route before adding it to the EIGRP topology table. EIGRP tests the SoO value for each route before sending updates to CE routers. Routes that are associated with SoO values that match the SoO value configured on the interface are filtered out before they are passed to the CE routers. When an EIGRP routing process receives routes that are associated with different SoO values, the SoO value is passed to the CE router and carried through the CE site.

Benefits of the EIGRP MPLS VPN PE-CE Site of Origin Support Feature

The configuration of the EIGRP MPLS VPN PE-CE Site of Origin Support feature introduces per-site VPN filtering, which improves support for complex topologies, such as MPLS VPNs with backdoor links, CE routers that are dual-homed to different PE routers, and PE routers that support CE routers from different sites within the same virtual routing and forwarding (VRF) instance.

How to Configure EIGRP MPLS VPN PE-CE Site of Origin Support

The following sections provide information about how to configure EIGRP MPLS VPN PE-CE Site of Origin Support:

Configuring the Site of Origin Extended Community

The configuration of the SoO extended community allows MPLS VPN traffic to be filtered on a per-site basis. The SoO extended community is configured in an inbound BGP route map on the PE router and is applied to the interface. The SoO extended community can be applied to all exit points at the customer site for more specific filtering but must be configured on all interfaces of PE routers that provide VPN services to CE routers.

Before you begin

- Confirm that the Border Gateway Protocol (BGP) is configured in the network core (or the service provider backbone).
- Configure an EIGRP MPLS VPN before configuring this feature.
- All PE routers that are configured to support the EIGRP MPLS VPN must support the SoO extended community.
- A unique SoO value must be configured for each VPN site. The same value must be used on the interface of the PE router that connects to the CE router for each VPN site.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**route-map** *map-name* {**permit** | **deny**} [*sequence-number*] **Example:** `Device(config)# route-map Site-of-Origin permit 10`	Enters route-map configuration mode and creates a route map. • The route map is created in this step so that SoO extended community can be applied.

Step 4	**set extcommunity soo***extended-community-value*
	Example:
	`Device(config-route-map)# set extcommunity soo 100:1`
		Sets BGP extended community attributes.
		• The soo keyword specifies the site of origin extended community attribute.
		• The extended-community-valueargument specifies the value to be set. The value can be one of the following formats:
		  • autonomous-system-number: network-number
		  • ip-address: network-number

	Command or Action	Purpose
		The colon is used to separate the autonomous system number and network number or IP address and network number.
Step 5	**exit** **Example:** `Device(config-route-map)# exit`	Exits route-map configuration mode and enters global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 1/0/1`	Enters interface configuration mode to configure the specified interface.
Step 7	**no switchport** **Example:** `Device(config-if)# no switchport`	causes the interface to cease operating as a Layer 2 port and become a Cisco-routed (Layer 3) port:
Step 8	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding VRF1`	Associates the VRF with an interface or subinterface. • The VRF name configured in this step should match the VRF name created for the EIGRP MPLS VPN with the MPLS VPN Support for EIGRP Between Provider Edge and Customer Edge feature.
Step 9	**ip vrf sitemap** *route-map-name* **Example:** `Device(config-if)# ip vrf sitemap Site-of-Origin`	Associates the VRF with an interface or subinterface. • The route map name configured in this step should match the route map name created to apply the SoO extended community in Step 3.
Step 10	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.0.0.1 255.255.255.255`	Configures the IP address for the interface. • The IP address needs to be reconfigured after enabling VRF forwarding.
Step 11	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and enters privileged EXEC mode.

What to do next

- For mixed EIGRP MPLS VPN network topologies that contain backdoor routes, the next task is to configure the "prebest path" cost community for backdoor routes.

Verifying the Configuration of the SoO Extended Community

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip bgp vpnv4** {**all** | **rd***route-distinguisher* | **vrf***vrf-name*} [*ip-prefix/length*] **Example:** `Device# ip bgp vpnv4 vrf SOO-1 20.2.1.1/32`	Displays VPN address information from the BGP table. • Use the show ip bgp vpnv4 command with the all keyword to verify that the specified route has been configured with the SoO extended community attribute.

Configuration Examples for EIGRP MPLS VPN PE-CE SoO

The following section shows configuration examples for EIGRP MPLS VPN PE-CE SoO:

Example Configuring the Site of Origin Extended Community

The following example, beginning in global configuration mode, configures SoO extended community on an interface:

```
 route-map Site-of-Origin permit 10
 set extcommunity soo 100:1
exit
GigabitEthernet1/0/1
vrf forwarding RED
ip vrf sitemap Site-of-Origin
ip address 10.0.0.1 255.255.255.255
end
```

Example Verifying the Site of Origin Extended Community

The following example shows VPN address information from the BGP table and verifies the configuration of the SoO extended community:

```
Device# show ip bgp vpnv4 all 10.0.0.1
 BGP routing table entry for 100:1:10.0.0.1/32, version 6
Paths: (1 available, best #1, no table)
Advertised to update-groups:
1
100 300
192.168.0.2 from 192.168.0.2 (172.16.13.13)
Origin incomplete, localpref 100, valid, external, best
Extended Community: SOO:100:1
```

Show command Customer Edge Device

```
Device# show ip eigrp topo 20.2.1.1/32
EIGRP-IPv4 Topology Entry for AS(30)/ID(30.0.0.1) for 20.2.1.1/32
  State is Passive, Query origin flag is 1, 2 Successor(s), FD is 131072
  Descriptor Blocks:
  31.1.1.2 (GigabitEthernet1/0/13), from 31.1.1.2, Send flag is 0x0
      Composite metric is (131072/130816), route is External
      Vector metric:
        Minimum bandwidth is 1000000 Kbit
        Total delay is 5020 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
        Originating router is 30.0.0.2
      Extended Community: SoO:100:1
      External data:
        AS number of route is 0
        External protocol is Connected, external metric is 0
        Administrator tag is 0 (0x00000000)
```

Show command Provider Edge Device

```
Device# show ip eigrp vrf SOO-1 topology 31.1.1.0/24
EIGRP-IPv4 VR(L3VPN) Topology Entry for AS(30)/ID(2.2.2.22)
           Topology(base) TID(0) VRF(SOO-1)
EIGRP-IPv4(30): Topology base(0) entry for 31.1.1.0/24
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 1310720
  Descriptor Blocks:
  1.1.1.1, from VPNv4 Sourced, Send flag is 0x0
      Composite metric is (1310720/0), route is Internal (VPNv4 Sourced)
      Vector metric:
        Minimum bandwidth is 1000000 Kbit
        Total delay is 10000000 picoseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 0
        Originating router is 1.1.1.11
      Extended Community: SoO:100:1
```

Feature History for EIGRP MPLS VPN PE-CE Site of Origin

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	EIGRP MPLS VPN PE-CE Site of Origin	The EIGRP MPLS VPN PE-CE Site of Origin feature introduces the capability to filter Multiprotocol Label Switching (MPLS) Virtual Private Network (VPN) traffic on a per-site basis for Enhanced Interior Gateway Routing Protocol (EIGRP) networks. Support for this feature was introduced only on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches.
Cisco IOS XE Gibraltar 16.11.1	EIGRP MPLS VPN PE-CE Site of Origin	Support for this feature was introduced only on the C9500-32C, C9500-32QC, C9500-48Y4C, and C9500-24Y4C models of the Cisco Catalyst 9500 Series Switches.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 4

Configuring Ethernet-over-MPLS and Pseudowire Redundancy

Configuring Ethernet-over-MPLS

This section provides information about how to configure Ethernet over Multiprotocol Label Switching (EoMPLS).

Information About EoMPLS

EoMPLS is one of the Any Transport over MPLS (AToM) transport types. EoMPLS works by encapsulating Ethernet protocol data units (PDUs) in MPLS packets and forwarding them across the MPLS network. Each PDU is transported as a single packet.

Only the following mode is supported:

- Port mode—Allows all traffic on a port to share a single virtual circuit across an MPLS network. Port mode uses virtual circuit type 5.

Prerequisites for Ethernet-over-MPLS

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

- Configure IP routing in the core so that the provider edge (PE) devices can reach each other through IP.
- Configure MPLS in the core so that a label switched path (LSP) exists between the PE devices.
- Configure the **no switchport**, **no keepalive**, and **no ip address** commands before configuring Xconnect on the attachment circuit.
- For load-balancing, configuring the **port-channel load-balance** command is mandatory.
- Subinterfaces must be supported to enable EoMPLS VLAN mode.

- The **mpls ldp graceful-restart** command must be configured to enable the device to protect LDP bindings and MPLS forwarding state during a disruption in service. We recommend you to configure this command (even if you do not want to preserve the forwarding state) to avoid device failure during SSO in a high availability setup with scale configurations.

Restrictions for EoMPLS

- VLAN mode is not supported. Ethernet Flow Point is not supported.
- QoS : Customer DSCP Re-marking is not supported with VPWS and EoMPLS.
- VCCV Ping with explicit null is not supported.
- L2 VPN Interworking is not supported.
- L2 Protocol Tunneling CLI is not supported.
- Untagged, tagged and 802.1Q in 802.1Q are supported as incoming traffic.

Note Flow Load balance for 802.1Q in 802.1Q over EoMPLS is not supported.

- Flow Aware Transport Pseudowire Redundancy (FAT PW) is supported only in Protocol-CLI mode. Supported load balancing parameters are Source IP, Source MAC address, Destination IP and Destination MAC address.
- Enabling or disabling Control word is supported.
- MPLS QoS is supported in Pipe and Uniform Mode. Default mode is Pipe Mode.
- Both – the legacy xconnect and Protocol-CLI (interface pseudowire configuration) modes are supported.
- Xconnect and MACSec cannot be configured on the same interface.
- MACSec should be configured on CE devices and Xconnect should be configured on PE devices.
- A MACSec session should be between CE devices.
- By default, EoMPLS PW tunnels all protocols like CDP, STP. EoMPLS PW cannot perform selective protocol tunneling as part of L2 Protocol Tunneling CLI.

Configuring Port-Mode EoMPLS

Port-Mode EoMPLS can be configured in two modes :

- Xconnect Mode
- Protocol CLI Method

Xconnect Mode

To configure EoMPLS port mode in Xconnect mode, perform the following task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface TenGigabitEthernet1/0/36`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Enters Layer 3 mode for physical ports only.
Step 5	**no ip address** **Example:** `Device(config-if)# no ip address`	Ensures that no IP address is assigned to the physical port.
Step 6	**no keepalive** **Example:** `Device(config-if)# no keepalive`	Ensures that the device does not send keepalive messages.
Step 7	**xconnect** *peer-device-id vc-id* **encapsulation mpls** **Example:** `Device(config-if)# xconnect 10.1.1.1 962 encapsulation mpls`	Binds the attachment circuit to a pseudowire virtual circuit (VC). The syntax for this command is the same as for all other Layer 2 transports.

	Command or Action	Purpose
Step 8	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Protocol CLI Method

To configure EoMPLS port mode in protocol CLI mode, perform the following task:

Procedure

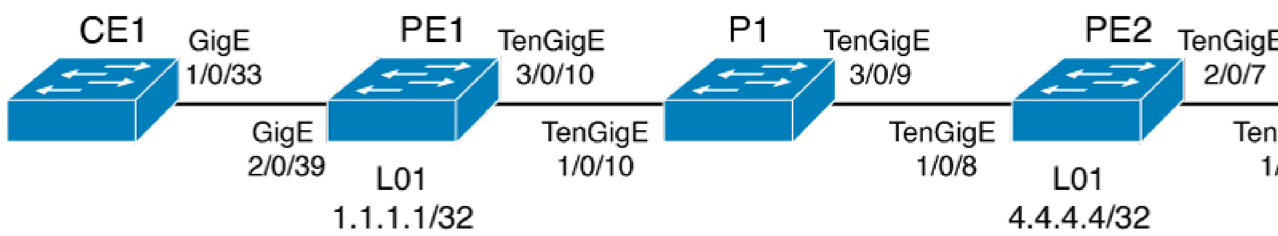
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**port-channel load-balance dst-ip** **Example:** `Device(config)# port-channel load-balance dst-ip`	Sets the load distribution method to the destination IP address.
Step 4	**interface** *interface-id* **Example:** `Device(config)# interface TenGigabitEthernet1/0/21`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 5	**no switchport** **Example:** `Device(config-if)# no switchport`	Enters Layer 3 mode for physical ports only.

	Command or Action	Purpose
Step 6	**no ip address** **Example:** `Device(config-if)# no ip address`	Ensures that no IP address is assigned to the physical port.
Step 7	**no keepalive** **Example:** `Device(config-if)# no keepalive`	Ensures that the device does not send keepalive messages.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 9	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 17`	Establishes a pseudowire interface with a value that you specify and enters pseudowire configuration mode.
Step 10	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies the tunneling encapsulation.
Step 11	**neighbor** *peer-ip-addr vc-id* **Example:** `Device(config-if)# neighbor 10.10.0.10 17`	Specifies the peer IP address and virtual circuit (VC) ID value of a Layer 2 VPN (L2VPN) pseudowire.
Step 12	**l2vpn xconnect context** *context-name* **Example:** `Device(config-if)# l2vpn xconnect context vpws17`	Creates an L2VPN cross connect context and enters Xconnect context configuration mode.

	Command or Action	Purpose
Step 13	**member** *interface-id* **Example:** `Device(config-if-xconn)# member TenGigabitEthernet1/0/21`	Specifies interface that forms an L2VPN cross connect.
Step 14	**member pseudowire** *number* **Example:** `Device(config-if-xconn)# member pseudowire 17`	Specifies the pseudowire interface that forms an L2VPN cross connect.
Step 15	**end** **Example:** `Device(config-if-xconn)# end`	Exits Xconnect interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for EoMPLS

Figure 2: EoMPLS Topology



<table>
<tr><th>PE Configuration</th><th>CE Configuration</th></tr>
<tr><td><pre>
mpls ip
mpls label protocol ldp
mpls ldp graceful-restart
mpls ldp router-id loopback 1 force
interface Loopback1
ip address 1.1.1.1 255.255.255.255

ip ospf 100 area 0
router ospf 100
router-id 1.1.1.1
nsf
system mtu 9198
port-channel load-balance dst-ip
!
interface GigabitEthernet2/0/39
no switchport
no ip address
no keepalive
!
interface pseudowire101
encapsulation mpls
neighbor 4.4.4.4 101
load-balance flow ip dst-ip
load-balance flow-label both
l2vpn xconnect context pw101
member pseudowire101
member GigabitEthernet2/0/39
!
interface TenGigabitEthernet3/0/10

switchport trunk allowed vlan 142
switchport mode trunk
channel-group 42 mode active
!
interface Port-channel42
switchport trunk allowed vlan 142
switchport mode trunk
!
interface Vlan142
ip address 142.1.1.1 255.255.255.0

ip ospf 100 area 0
mpls ip
mpls label protocol ldp
!
</pre></td><td><pre>
interface GigabitEthernet1/0/33
switchport trunk allowed vlan 912
switchport mode trunk spanning-tree portfast trunk
!
interface Vlan912
ip address 10.91.2.3 255.255.255.0
!
</pre></td></tr>
</table>

The following is a sample output of **show mpls l2 vc vcid vc-id detail** command:

```
Local interface: Gi1/0/1 up, line protocol up, Ethernet up
 Destination address: 1.1.1.1, VC ID: 101, VC status: up
Output interface: Vl182, imposed label stack {17 16}
Preferred path: not configured
Default path: active
Next hop: 182.1.1.1
Load Balance: ECMP
flow classification: ip dst-ip
Create time: 06:22:11, last status change time: 05:58:42
Last label FSM state change time: 05:58:42  Signaling protocol:
LDP, peer 1.1.1.1:0 up
Targeted Hello: 4.4.4.4(LDP Id) -> 1.1.1.1, LDP is UP
```

```
Graceful restart: not configured and not enabled
Non stop routing: not configured and not enabled
Status TLV support (local/remote)   : enabled/supported
LDP route watch                   : enabled
Label/status state machine        : established, LruRru
Last local dataplane   status rcvd: No fault
Last BFD dataplane     status rcvd: Not sent
Last BFD peer monitor  status rcvd: No fault
Last local AC  circuit status rcvd: No fault
Last local AC  circuit status sent: No fault
Last local PW i/f circ status rcvd: No fault
Last local LDP TLV     status sent: No fault
Last remote LDP TLV    status rcvd: No fault
Last remote LDP ADJ    status rcvd: No fault
MPLS VC labels: local 512, remote 16
Group ID: local n/a, remote 0
MTU: local 9198, remote 9198
Remote interface description:   Sequencing: receive disabled, send disabled
Control Word: On (configured: autosense)
SSO Descriptor: 1.1.1.1/101, local label: 512
Dataplane:
SSM segment/switch IDs: 4096/4096 (used), PWID: 1
VC statistics:    transit packet totals: receive 172116845, send 172105364
transit byte totals:   receive 176837217071, send 172103349728
 transit packet drops:  receive 0, seq error 0, send 0
```

The following is a sample output of **show l2vpn atom vc vcid vc-id detail** command:

```
pseudowire101 is up, VC status is up PW type: Ethernet
Create time: 06:30:41, last status change time: 06:07:12
Last label FSM state change time: 06:07:12
Destination address: 1.1.1.1 VC ID: 101
Output interface: Vl182, imposed label stack {17 16}
 Preferred path: not configured
 Default path: active    Next hop: 182.1.1.1
Load Balance: ECMP  Flow classification: ip dst-ip
Member of xconnect service pw101
Associated member Gi1/0/1 is up, status is up
Interworking type is Like2Like    Service id: 0xe5000001
Signaling protocol: LDP, peer 1.1.1.1:0 up
Targeted Hello: 4.4.4.4(LDP Id) -> 1.1.1.1, LDP is UP
Graceful restart: not configured and not enabled
Non stop routing: not configured and not enabled
PWid FEC (128), VC ID: 101    Status TLV support (local/remote)         : enabled/supported

 LDP route watch                          : enabled
Label/status state machine               : established, LruRru
Local dataplane status received          : No fault
BFD dataplane status received            : Not sent
BFD peer monitor status received         : No fault
Status received from access circuit      : No fault
Status sent to access circuit            : No fault
Status received from pseudowire i/f      : No fault
Status sent to network peer              : No fault
Status received from network peer        : No fault
Adjacency status of remote peer          : No fault
Sequencing: receive disabled, send disabled  Bindings
 Parameter    Local                         Remote
------------ ------------------------------ ------------------------------
Label        512                           16
Group ID     n/a                           0
Interface
MTU          9198                          9198
Control word on (configured: autosense)     on
```

```
PW type      Ethernet                       Ethernet
VCCV CV type 0x02                           0x02
                   LSPV [2]                       LSPV [2]
VCCV CC type 0x06                           0x06
                   RA [2], TTL [3]               RA [2], TTL [3]
Status TLV   enabled                        supported
Flow Label   T=1, R=1                       T=1, R=1
SSO Descriptor: 1.1.1.1/101, local label: 512
Dataplane:
SSM segment/switch IDs: 4096/4096 (used), PWID: 1
Rx Counters    176196691 input transit packets, 181028952597 bytes
 0 drops, 0 seq err
Tx Counters    176184928 output transit packets, 176182865992 bytes
0 drops
```

The following is a sample output of **show mpls forwarding-table** command:

```
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
57         18         1.1.1.1/32       0             Po45       145.1.1.1
           No Label   1.1.1.1/32       0             Te1/0/2    147.1.1.1
           No Label   1.1.1.1/32       0             Te1/0/11   149.1.1.1
           No Label   1.1.1.1/32       0             Te1/0/40   155.1.1.1
```

Configuring Pseudowire Redundancy

This section provides information about how to configure pseudowire redundancy.

Overview of Pseudowire Redundancy

The L2VPN pseudowire redundancy feature enables you to configure your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. This feature provides the ability to recover from a failure either of the remote provider edge (PE) device or of the link between the PE and customer edge (CE) devices.

Pseudowire redundancy can be configured using both the Xconnect and the protocol CLI method.

Prerequisites for Pseudowire Redundancy

- Configure the **no switchport**, **no keepalive**, and **no ip address** before configuring Xconnect mode to connect the attachment circuit.
- For load-balancing, configure the **port-channel load-balance** command.
- Subinterfaces must be supported to enable pseudowire redundancy VLAN mode.

Restrictions for Pseudowire Redundancy

- VLAN mode, EFP (Ethernet Flow Point) and IGMP Snooping is not supported.
- PWR is supported with port mode EoMPLS only.
- Untagged, tagged and 802.1Q in 802.1Q are supported as incoming traffic.

Note Load balance for 802.1Q in 802.1Q with Pseudowire Redundancy is not supported.

- Flow Label for ECMP Load balancing in core network based on customer's source IP, destination IP, source MAC and destination MAC.
- Enabling or disabling Control word is supported.
- MPLS QoS is supported in Pipe and Uniform Mode. Default mode is Pipe Mode.
- Port-channel as attachment circuit is not supported.
- QoS : Customer DSCP Re-marking is not supported with VPWS and EoMPLS.
- VCCV Ping with explicit null is not supported.
- L2 VPN Interworking is not supported.
- **ip unnumbered** command is not supported in MPLS configuration.
- Not more than one backup pseudowire supported.
- PW redundancy group switchover is not supported

Configuring Pseudowire Redundancy

Pseudowire Redundancy can be configured in two modes :

- Xconnect Mode
- Protocol CLI Method

Xconnect Mode

To configure pseudowire redundancy port mode in Xconnect mode, perform the following task:

Note To enable load balance, use the corresponding **load-balance** commands from Xconnect Mode procedure of the 'How to Configure Ethernet-over-MPLS section.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface GigabitEthernet1/0/44`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Enters Layer 3 mode, for physical ports only.
Step 5	**no ip address** **Example:** `Device(config-if)# no ip address`	Ensures that there is no IP address assigned to the physical port.
Step 6	**no keepalive** **Example:** `Device(config-if)# no keepalive`	Ensures that the device does not send keepalive messages.
Step 7	**xconnect** *peer-device-id vc-id* **encapsulation mpls** **Example:** `Device(config-if)# xconnect 10.1.1.1 117 encapsulation mpls`	Binds the attachment circuit to a pseudowire VC. The syntax for this command is the same as for all other Layer 2 transports.
Step 8	**backup peer** *peer-router-ip-addr* **vcid** *vc-id* [**priority** *value*] **Example:** `Device(config-if)# backup peer`	Specifies a redundant peer for a pseudowire VC.

	Command or Action	Purpose
	10.11.11.11 118 priority 9	
Step 9	**end** **Example:** `Device(config)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Protocol CLI Method

To configure pseudowire redundancy port mode in protocol CLI mode, perform the following task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**port-channel load-balance dst-ip** **Example:** `Device(config)# port-channel load-balance dst-ip`	Sets the load-distribution method to the destination IP address.
Step 4	**interface** *interface-id* **Example:** `Device(config)# interface TenGigabitEthernet1/0/36`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 5	**no switchport** **Example:** `Device(config-if)# no switchport`	Enters Layer 3 mode, for physical ports only.

	Command or Action	Purpose
Step 6	**no ip address** **Example:** `Device(config-if)# no ip address`	Ensures that there is no IP address assigned to the physical port.
Step 7	**no keepalive** **Example:** `Device(config-if)# no keepalive`	Ensures that the device does not send keepalive messages.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 9	**interface pseudowire** *number-active* **Example:** `Device(config)# interface pseudowire 17`	Establishes an active pseudowire interface with a value that you specify and enters pseudowire configuration mode.
Step 10	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies the tunneling encapsulation.
Step 11	**neighbor** *active-peer-ip-addr vc-id* **Example:** `Device(config-if)# neighbor 10.10.0.10 17`	Specifies the active peer IP address and VC ID value of a L2VPN pseudowire.
Step 12	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 13	**interface pseudowire** *number-standby* **Example:** `Device(config)# interface pseudowire 18`	Establishes a standby pseudowire interface with a value that you specify and enters pseudowire configuration mode.
Step 14	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies the tunneling encapsulation.
Step 15	**neighbor** *standby-peer-ip-addr vc-id* **Example:** `Device(config-if)# neighbor 10.10.0.11 18`	Specifies the standby peer IP address and VC ID value of a L2VPN pseudowire.
Step 16	**l2vpn xconnect context** *context-name* **Example:** `Device(config-if)# l2vpn xconnect context vpws17`	Creates a L2VPN cross connect context, and attaches the VLAN mode EoMPLS attachment circuit to the active and standby pseudowire interfaces.
Step 17	**member** *interface-id* **Example:** `Device(config-if-xconn)# member TenGigabitEthernet1/0/36`	Specifies interface that forms a L2VPN cross connect.
Step 18	**member pseudowire** *number-active* **group** *group-name* [**priority** *value*] **Example:** `Device(config-if-xconn)# member pseudowire 17 group pwr10`	Specifies active pseudowire interface that forms a L2VPN cross connect.
Step 19	**member pseudowire** *number-standby* **group** *group-name* [**priority** *value*] **Example:**	Specifies standby pseudowire interface that forms a L2VPN cross connect.

	Command or Action	Purpose
	Device(config-if-xconn)# **member pseudowire 18 group pwr10 priority 6**	
Step 20	**end** **Example:** Device(config-if-xconn)# **end**	Exits Xconnect configuration mode and returns to privileged EXEC mode.

Configuration Examples for Pseudowire Redundancy

<table>
<tr><th>PE Configuration</th><th>CE Configuration</th></tr>
<tr><td><pre>
mpls ip
mpls label protocol ldp
mpls ldp graceful-restart
mpls ldp router-id loopback 1 force
!
interface Loopback1
ip address 1.1.1.1 255.255.255.255
ip ospf 100 area 0
router ospf 100
router-id 1.1.1.1
nsf
!
interface GigabitEthernet2/0/39
no switchport
no ip address
no keepalive
!
interface pseudowire101
encapsulation mpls
neighbor 4.4.4.4 101
!
interface pseudowire102
encapsulation mpls
neighbor 3.3.3.3 101
l2vpn xconnect context pw101
member pseudowire101 group pwgrp1 priority 1
member pseudowire102 group pwgrp1 priority 15

member GigabitEthernet2/0/39
!
interface TenGigabitEthernet3/0/10
switchport trunk allowed vlan 142
switchport mode trunk
channel-group 42 mode active
!
interface Port-channel42
switchport trunk allowed vlan 142
switchport mode trunk
!
interface Vlan142
ip address 142.1.1.1 255.255.255.0
ip ospf 100 area 0
mpls ip
mpls label protocol ldp
!
</pre></td><td><pre>
interface GigabitEthernet1/0/33
switchport trunk allowed vlan 912
switchport mode trunk spanning-tree portfast
 trunk
!
interface Vlan912
ip address 10.91.2.3 255.255.255.0
!
</pre></td></tr>
</table>

The following is sample output of the **show mpls l2transport vc** *vc-id* command :

```
Device# show mpls l2transport vc 101
Local intf     Local circuit              Dest address        VC ID      Status
------------- -------------------------- ---------------      ---------- ----------
Gi2/0/39       Ethernet                   4.4.4.4             101        UP


Device# show mpls l2transport vc 102
Local intf    Local circuit               Dest address        VC ID      Status
------------- -------------------------- ---------------      ---------- ----------
```

```
Gi2/0/39      Ethernet                    3.3.3.3             102        STANDBY
```

Feature History for Ethernet-over-MPLS and Pseudowire Redundancy

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	Ethernet-over-MPLS and Pseudowire Redundancy	Ethernet-over-MPLS is one of the Any Transport over MPLS (AToM) transport types. EoMPLS works by encapsulating Ethernet protocol data units (PDUs) in MPLS packets and forwarding them across the MPLS network. Each PDU is transported as a single packet. The L2VPN pseudowire redundancy feature enables you to configure your network to detect a failure in the network and reroute the Layer 2 service to another endpoint that can continue to provide service. Port mode support is introduced. Support for this feature was introduced only on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches.
Cisco IOS XE Fuji 16.9.1	Ethernet-over-MPLS and Pseudowire Redundancy	Support for this feature was introduced only on the C9500-32C, C9500-32QC, C9500-48Y4C, and C9500-24Y4C models of the Cisco Catalyst 9500 Series Switches.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/

http://www.cisco.com/go/cfn.

CHAPTER 5

Configuring IPv6 Provider Edge over MPLS (6PE)

Prerequisites for 6PE

Redistribute PE-CE IGP IPv6 routes into core BGP and vice-versa

Restrictions for 6PE

eBGP as CE-PE is not supported. Static Routes, OSPFv3, ISIS, RIPv2 are supported as CE-PE.

Information About 6PE

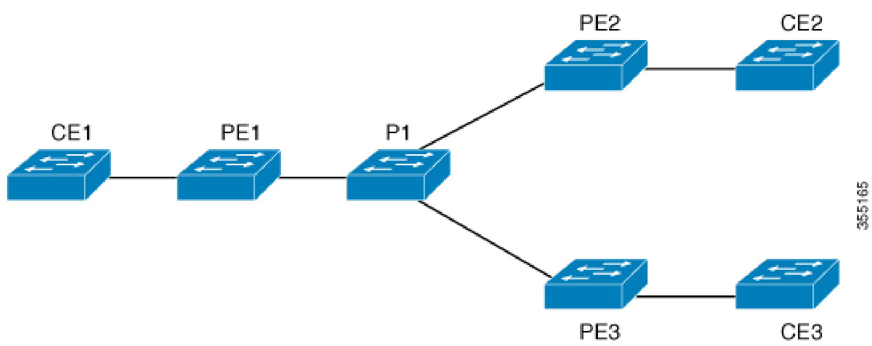
6PE is a technique that provides global IPv6 reachability over IPv4 MPLS. It allows one shared routing table for all other devices. 6PE allows IPv6 domains to communicate with one another over the IPv4 without an explicit tunnel setup, requiring only one IPv4 address per IPv6 domain.

While implementing 6PE, the provider edge routers are upgraded to support 6PE, while the rest of the core network is not touched (IPv6 unaware). This implementation requires no reconfiguration of core routers because forwarding is based on labels rather than on the IP header itself. This provides a cost-effective strategy for deploying IPv6.The IPv6 reachability information is exchanged by PE routers using multiprotocol Border Gateway Protocol (mp-iBGP) extensions.

6PE relies on mp-iBGP extensions in the IPv4 network configuration on the PE router to exchange IPv6 reachability information in addition to an MPLS label for each IPv6 address prefix to be advertised. PE routers are configured as dual stacks, running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange. The next hop advertised by the PE router for 6PE and 6VPE prefixes is still the IPv4 address that is used for IPv4 L3 VPN routes. A value of ::FFFF: is prepended to the IPv4 next hop, which is an IPv4-mapped IPv6 address.

The following figure illustrates the 6PE topology.

Figure 3: 6PE Topology



Configuring 6PE

Ensure that you configure 6PE on PE routers participating in both the IPv4 cloud and IPv6 clouds.

BGP running on a PE router should establish (IPv4) neighborhood with BGP running on other PEs. Subsequently, it should advertise the IPv6 prefixes learnt from the IPv6 table to the neighbors. The IPv6 prefixes advertised by BGP would automatically have IPv4-encoded-IPv6 addresses as the nexthop-address in the advertisement.

To configure 6PE, complete the following steps:

Procedure

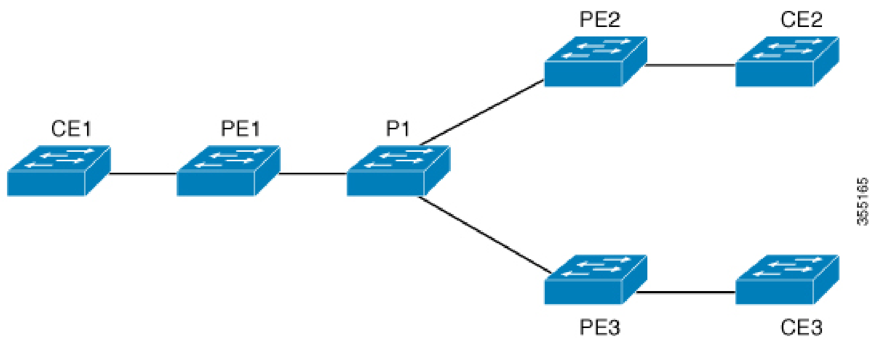
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.

	Command or Action	Purpose
Step 4	**router bgp** *as-number* **Example:** `Device(config)# router bgp 65001`	Enters the number that identifies the autonomous system (AS) in which the router resides. *as-number*—Autonomous system number. Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.
Step 5	**bgp router-id interface** *interface-id* **Example:** `Device(config-router)# bgp router-id interface Loopback1`	Configures a fixed router ID for the local Border Gateway Protocol (BGP) routing process.
Step 6	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 7	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-restart`	Enables the Border Gateway Protocol (BGP) graceful restart capability globally for all BGP neighbors.
Step 8	**neighbor** { *ip-address* | *ipv6-address* | *peer-group-name* } **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 33.33.33.33 remote-as 65001`	Adds an entry to the BGP or multiprotocol BGP neighbor table. • *ip-address*—IP address of a peer router with which routing information will be exchanged. • *ipv6-address*—IPv6 address of a peer router with which routing information will be exchanged. • *peer-group-name*—Name of the BGP peer group. • *remote-as*—Specifies a remote autonomous system. • *as-number*—Number of an autonomous system to which the neighbor belongs, ranging from 1 to 65535.
Step 9	**neighbor** { *ip-address* | *ipv6-address* | *peer-group-name* } **update-source** *interface-type interface-number* **Example:**	Configures BGP sessions to use any operational interface for TCP connections.

	Command or Action	Purpose
	`Device(config-router)# neighbor 33.33.33.33 update-source Loopback1`	
Step 10	**address-family ipv6** **Example:** `Device(config-router)# address-family ipv6`	Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard IPv6 address prefixes.
Step 11	**redistribute** *protocol as-number* **match** { **internal** | **external 1** | **external 2** **Example:** `Device(config-router-af)# redistribute ospf 11 match internal external 1`	Redistributes routes from one routing domain into another routing domain.
Step 12	**neighbor** { *ip-address* | *ipv6-address* | *peer-group-name* } **activate** **Example:** `Device(config-router-af)# neighbor 33.33.33.33 activate`	Enables the exchange of information with a BGP neighbor.
Step 13	**neighbor** { *ip-address* | *ipv6-address* | *peer-group-name* } **send-label** **Example:** `Device(config-router-af)#  neighbor 33.33.33.33 send-label`	Sends MPLS labels with BGP routes to a neighboring BGP router.
Step 14	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits BGP address-family submode.
Step 15	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuration Examples for 6PE

Figure 4: 6PE Topology



PE Configuration

`router ospfv3 11`
`ip routing`
`ipv6 unicast-routing`
`address-family ipv6 unicast`
`redistribute bgp 65001`
`exit-address-family`
`!`
`router bgp 65001`
`bgp router-id interface Loopback1`
`bgp log-neighbor-changes`
`bgp graceful-restart`
`neighbor 33.33.33.33 remote-as 65001`
`neighbor 33.33.33.33 update-source Loopback1`
`!`
`address-family ipv4`
`neighbor 33.33.33.33 activate`
`!`
`address-family ipv6`
`redistribute ospf 11 match internal external 1 external 2 include-connected`
`neighbor 33.33.33.33 activate`
`neighbor 33.33.33.33 send-label`
`neighbor 33.33.33.33 send-community extended`
`!`

The following is a sample output of **show bgp ipv6 unicast summary** :

```
BGP router identifier 1.1.1.1, local AS number 100
BGP table version is 34, main routing table version 34
4 network entries using 1088 bytes of memory
4 path entries using 608 bytes of memory
4/4 BGP path/bestpath attribute entries using 1120 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 2816 total bytes of memory
BGP activity 6/2 prefixes, 16/12 paths, scan interval 60 secs
```

```
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
  State/PfxRcd
2.2.2.2         4          100      21      21       34    0    0 00:04:57
        2

sh ipv route
IPv6 Routing Table - default - 7 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, R - RIP, I1 - ISIS L1, I2 - ISIS L2
       IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP
external
      ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect

       RL - RPL, O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1
       OE2 - OSPF ext 2, ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2
       la - LISP alt, lr - LISP site-registrations, ld - LISP dyn-eid lA
 - LISP away
C   10:1:1:2::/64 [0/0]
     via Vlan4, directly connected
L   10:1:1:2::1/128 [0/0]
     via Vlan4, receive
LC  11:11:11:11::11/128 [0/0]
     via Loopback1, receive
B   30:1:1:2::/64 [200/0]
     via 33.33.33.33%default, indirectly connected
B   40:1:1:2::/64 [200/0]
     via 44.44.44.44%default, indirectly connected
```

The following is a sample output of **show bgp ipv6 unicast** command :

```
BGP table version is 112, local router ID is 11.11.11.11
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale, m multipath, b backup-path, f
RT-Filter,
              x best-external, a additional-path, c RIB-compressed,
              t secondary path,
Origin codes: i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found
     Network          Next Hop            Metric LocPrf Weight Path
 *>   10:1:1:2::/64    ::                       0         32768 ?
 *>i  30:1:1:2::/64    ::FFFF:33.33.33.33
                                                0    100      0 ?
 *>i  40:1:1:2::/64    ::FFFF:44.44.44.44
                                                0    100      0 ?
 *>i  173:1:1:2::/64   ::FFFF:33.33.33.33
                                                2    100      0 ?
```

The following is a sample output of **show ipv6 cef 40:1:1:2::0/64 detail** command :

```
40:1:1:2::/64, epoch 6, flags [rib defined all labels]
  recursive via 44.44.44.44 label 67
    nexthop 1.20.4.2 Port-channel103 label 99-(local:147)
```

Feature History for IPv6 Provider Edge over MPLS (6PE)

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	IPv6 Provider Edge over MPLS (6PE)	IPv6 Provider Edge over MPLS (6PE) provides global IPv6 reachability over IPv4 MPLS and allows one shared routing table for all other devices. Support for this feature was introduced only on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches.
Cisco IOS XE Fuji 16.9.1	IPv6 Provider Edge over MPLS (6PE)	Support for this feature was introduced on the C9500-32C, C9500-32QC, C9500-48Y4C, and C9500-24Y4C models of the Cisco Catalyst 9500 Series Switches.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 6

Configuring IPv6 VPN Provider Edge over MPLS (6VPE)

Configuring 6VPE

This section provides information about Configuring 6VPE on the switch.

Restrictions for 6VPE

- Inter-AS and carrier supporting carrier (CSC) is not supported.
- VRF Route-Leaking is not supported.
- eBGP as CE-PE is not supported.
- EIGRP, OSPFv3, RIP, ISIS, Static Routes are supported as CE-PE.
- MPLS Label Allocation modes supported are Per-VRF and Per-Prefix. Per-Prefix is the default mode.
- IP fragmentation is not supported in the Per-Prefix mode of Layer 3 VPN.
- DHCPv6 is not supported on a 6VPE topology with per-port trust enabled.

Information About 6VPE

6VPE is a mechanism to use the IPv4 backbone to provide VPN IPv6 services. It takes advantage of operational IPv4 MPLS backbones, eliminating the need for dual-stacking within the MPLS core. This translates to savings in operational costs and addresses the security limitations of the 6PE approach. 6VPE is more like a regular IPv4 MPLS-VPN provider edge, with an addition of IPv6 support within VRF. It provides logically separate routing table entries for VPN member devices.

Components of MPLS-based 6VPE Network

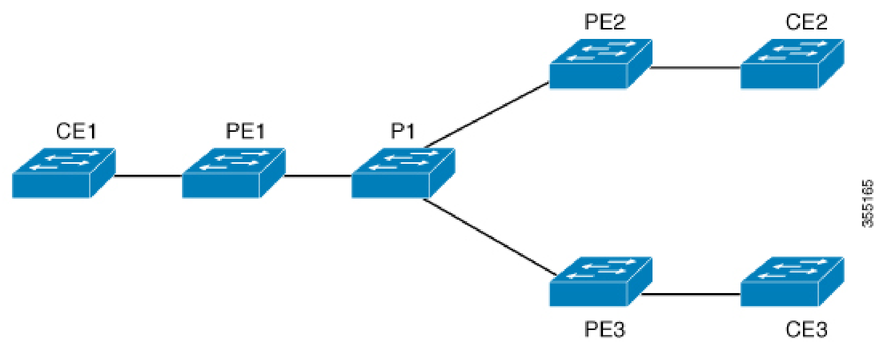
- VPN route target communities – A list of all other members of a VPN community.

- Multiprotocol BGP (MP-BGP) peering of VPN community PE routers – Propagates VRF reachability information to all members of a VPN community.
- MPLS forwarding – Transports all traffic between all VPN community members across a VPN service-provider network.

In the MPLS-VPN model a VPN is defined as a collection of sites sharing a common routing table. A customer site is connected to the service provider network by one or more interfaces, where the service provider associates each interface with a VPN routing table–known as the VRF table.

Configuration Examples for 6VPE

Figure 5: 6VPE Topology



PE Configuration

PE Configuration	C

```
vrf definition 6VPE-1
 rd 65001:11
 route-target export 1:1
 route-target import 1:1
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
interface TenGigabitEthernet1/0/38
 no switchport
 vrf forwarding 6VPE-1
 ip address 10.3.1.1 255.255.255.0
 ip ospf 2 area 0
 ipv6 address 10:111:111:111::1/64
 ipv6 enable
 ospfv3 1 ipv6 area 0
!
router ospf 2 vrf 6VPE-1
 router-id 1.1.11.11
 redistribute bgp 65001 subnets
!
router ospfv3 1
nsr
graceful-restart
!
address-family ipv6 unicast vrf 6VPE-1
redistribute bgp 65001
exit-address-family
!
router bgp 65001
bgp router-id interface Loopback1
bgp log-neighbor-changes
bgp graceful-restart
neighbor 33.33.33.33 remote-as 65001
neighbor 33.33.33.33 update-source Loopback1
!
 address-family ipv4 vrf 6VPE-1
  redistribute ospf 2 match internal external 1 external 2
 exit-address-family
 address-family ipv6 vrf 6VPE-1
  redistribute ospf 1 match internal external 1 external 2 include-connected
 exit-address-family
!
address-family vpnv4
neighbor 33.33.33.33 activate
neighbor 33.33.33.33 send-community both
neighbor 44.44.44.44 activate
neighbor 44.44.44.44 send-community both
neighbor 55.55.55.55 activate
neighbor 55.55.55.55 send-community both
exit-address-family
!
address-family vpnv6
neighbor 33.33.33.33 activate
neighbor 33.33.33.33 send-community both
neighbor 44.44.44.44 activate
neighbor 44.44.44.44 send-community both
neighbor 55.55.55.55 activate
```

PE Configuration
neighbor 55.55.55.55 send-community both exit-address-family !

The following is a sample output of **show mpls forwarding-table vrf** :

```
Local Outgoing Prefix Bytes Label Outgoing Next Hop
Label Label or Tunnel Id Switched interface
29 No Label A:A:A:565::/64[V] \ 0 aggregate/VRF601
32 No Label A:B5:1:5::/64[V] 2474160 Vl601 FE80::200:7BFF:FE62:2636
33 No Label A:B5:1:4::/64[V] 2477978 Vl601 FE80::200:7BFF:FE62:2636
35 No Label A:B5:1:3::/64[V] 2477442 Vl601 FE80::200:7BFF:FE62:2636
36 No Label A:B5:1:2::/64[V] 2476906 Vl601 FE80::200:7BFF:FE62:2636
37 No Label A:B5:1:1::/64[V] 2476370 Vl601 FE80::200:7BFF:FE62:2636
```

The following is a sample output of **show vrf counter** command :

```
Maximum number of VRFs supported: 256
Maximum number of IPv4 VRFs supported: 256
Maximum number of IPv6 VRFs supported: 256
Maximum number of platform iVRFs supported: 10
Current number of VRFs: 127
Current number of IPv4 VRFs: 6
Current number of IPv6 VRFs: 127
Current number of VRFs in delete state: 0
Current number of platform iVRFs: 1
```

The following is a sample output of **show ipv6 route vrf** command :

```
IPv6 Routing Table - VRF1 - 8 entries Codes: C - Connected, L - Local, S
 - Static, U - Per-user Static route B - BGP, R - RIP, I1 - ISIS L1, I2
- ISIS L2 IA - ISIS interarea, IS - ISIS summary, D - EIGRP, EX - EIGRP
external ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr -
Redirect RL - RPL, O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1 OE2
 - OSPF ext 2, ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2 la - LISP
alt, lr - LISP site-registrations, ld - LISP dyn-eid lA - LISP away

B 1:1:1:1::1/128 [200/1] via 1.1.1.11%default, indirectly connected
O 2:2:2:2::2/128 [110/1] via FE80::A2E0:AFFF:FE30:3E40,
TenGigabitEthernet1/0/7
B 3:3:3:3::3/128 [200/1] via 3.3.3.33%default, indirectly connected
B 10:1:1:1::/64 [200/0] via 1.1.1.11%default, indirectly connected
C 10:2:2:2::/64 [0/0] via TenGigabitEthernet1/0/7, directly connected
L 10:2:2:2::1/128 [0/0] via TenGigabitEthernet1/0/7, receive
B 10:3:3:3::/64 [200/0] via 3.3.3.33%default, indirectly connected
L FF00::/8 [0/0] via Null0, receive
```

Feature History for IPv6 VPN Provider Edge over MPLS (6VPE)

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	IPv6 VPN Provider Edge over MPLS (6VPE)	IPv6 VPN Provider Edge over MPLS (6VPE) is a mechanism to use the IPv4 backbone to provide VPN IPv6 services. It takes advantage of operational IPv4 MPLS backbones, eliminating the need for dual-stacking within the MPLS core. Support for this feature was introduced only on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 7

Configuring MPLS Layer 3 VPN

An MPLS Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of a Multiprotocol Label Switching (MPLS) provider core network. At each customer site, one or more customer edge (CE) devices attach to one or more provider edge (PE) devices. This module explains how to create an MPLS Layer 3 VPN.

MPLS Layer 3 VPNs

An MPLS Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of a Multiprotocol Label Switching (MPLS) provider core network. At each customer site, one or more customer edge (CE) devices attach to one or more provider edge (PE) devices. This module explains how to create an MPLS VPN.

Prerequisites for MPLS Virtual Private Networks

- Make sure that you have installed Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP), and Cisco Express Forwarding in your network.
- All devices in the core, including the provider edge (PE) devices, must be able to support Cisco Express Forwarding and MPLS forwarding. See the “Assessing the Needs of the MPLS Virtual Private Network Customers” section.
- Enable Cisco Express Forwarding on all devices in the core, including the PE devices. For information about how to determine if Cisco Express Forwarding is enabled, see the “Configuring Basic Cisco Express Forwarding” module in the *Cisco Express Forwarding Configuration Guide*.
- The **mpls ldp graceful-restart** command must be configured to enable the device to protect LDP bindings and MPLS forwarding state during a disruption in service. We recommend you to configure this command (even if you do not want to preserve the forwarding state) to avoid device failure during SSO in a high availability setup with scale configurations.

Restrictions for MPLS Virtual Private Networks

When static routes are configured in a Multiprotocol Label Switching (MPLS) or MPLS virtual private network (VPN) environment, some variations of the **ip route** and **ip route vrf** commands are not supported. Use the following guidelines when configuring static routes.

Supported Static Routes in an MPLS Environment

The following **ip route** command is supported when you configure static routes in an MPLS environment:

- **ip route** *destination-prefix mask interface next-hop-address*

The following **ip route** commands are supported when you configure static routes in an MPLS environment and configure load sharing with static nonrecursive routes and a specific outbound interface:

- **ip route** *destination-prefix mask* **interface1 next-hop1**
- **ip route** *destination-prefix mask* **interface2 next-hop2**

Unsupported Static Routes in an MPLS Environment That Uses the TFIB

The following **ip route** command is not supported when you configure static routes in an MPLS environment:

- **ip route** *destination-prefix mask next-hop-address*

The following **ip route** command is not supported when you configure static routes in an MPLS environment and enable load sharing where the next hop can be reached through two paths:

- **ip route** *destination-prefix mask next-hop-address*

The following **ip route** commands are not supported when you configure static routes in an MPLS environment and enable load sharing where the destination can be reached through two next hops:

- **ip route** *destination-prefix mask* **next-hop1**
- **ip route** *destination-prefix mask* **next-hop2**

Use the *interface* an *next-hop* arguments when specifying static routes.

Supported Static Routes in an MPLS VPN Environment

The following **ip route vrf** commands are supported when you configure static routes in an MPLS VPN environment, and the next hop and interface are in the same VRF:

- **ip route vrf** *vrf-name destination-prefix mask next-hop-address*
- **ip route vrf** *vrf-name destination-prefix mask interface next-hop-address*
- **ip route vrf** *vrf-name destination-prefix mask* **interface1 next-hop1**
- **ip route vrf** *vrf-name destination-prefix mask* **interface2 next-hop2**

The following **ip route vrf** commands are supported when you configure static routes in an MPLS VPN environment, and the next hop is in the global table in the MPLS cloud in the global routing table. For example, these commands are supported when the next hop is pointing to the Internet gateway.

- **ip route vrf** *vrf-name destination-prefix mask next-hop-address* **global**

- **ip route vrf** *vrf-name destination-prefix mask interface next-hop-address* (This command is supported when the next hop and interface are in the core.)

The following **ip route** commands are supported when you configure static routes in an MPLS VPN environment and enable load sharing with static nonrecursive routes and a specific outbound interface:

- **ip route** *destination-prefix mask* **interface1 next-hop1**
- **ip route** *destination-prefix mask* **interface2 next-hop2**

Unsupported Static Routes in an MPLS VPN Environment That Uses the TFIB

The following **ip route** command is not supported when you configure static routes in an MPLS VPN environment, the next hop is in the global table in the MPLS cloud within the core, and you enable load sharing where the next hop can be reached through two paths:

- **ip route vrf** *destination-prefix mask next-hop-address* **global**

The following **ip route** commands are not supported when you configure static routes in an MPLS VPN environment, the next hop is in the global table in the MPLS cloud within the core, and you enable load sharing where the destination can be reached through two next hops:

- **ip route vrf** *destination-prefix mask* **next-hop1 global**
- **ip route vrf** *destination-prefix mask* **next-hop2 global**

The following **ip route vrf** commands are not supported when you configure static routes in an MPLS VPN environment, and the next hop and interface are in the same VRF:

- **ip route vrf** *vrf-name destination-prefix mask* **next-hop1** *vrf-name destination-prefix mask* **next-hop1**
- **ip route vrf** *vrf-name destination-prefix mask* **next-hop2**

Supported Static Routes in an MPLS VPN Environment Where the Next Hop Resides in the Global Table on the CE Device

The following **ip route vrf** command is supported when you configure static routes in an MPLS VPN environment, and the next hop is in the global table on the customer edge (CE) side. For example, the following command is supported when the destination prefix is the CE device's loopback address, as in external Border Gateway Protocol (EBGP) multihop cases.

- **ip route vrf** *vrf-name destination-prefix mask interface next-hop-address*

The following **ip route** commands are supported when you configure static routes in an MPLS VPN environment, the next hop is in the global table on the CE side, and you enable load sharing with static nonrecursive routes and a specific outbound interface:

- **ip route** *destination-prefix mask* **interface1 nexthop1**
- **ip route** *destination-prefix mask* **interface2 nexthop2**

Information About MPLS Virtual Private Networks

This section provides information about MPLS Virtual Private Networks:

MPLS Virtual Private Network Definition

Before defining a Multiprotocol Label Switching virtual private network (MPLS VPN), you must define a VPN in general. A VPN is:

- An IP-based network delivering private network services over a public infrastructure
- A set of sites that 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, because 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 the customer's involvement.

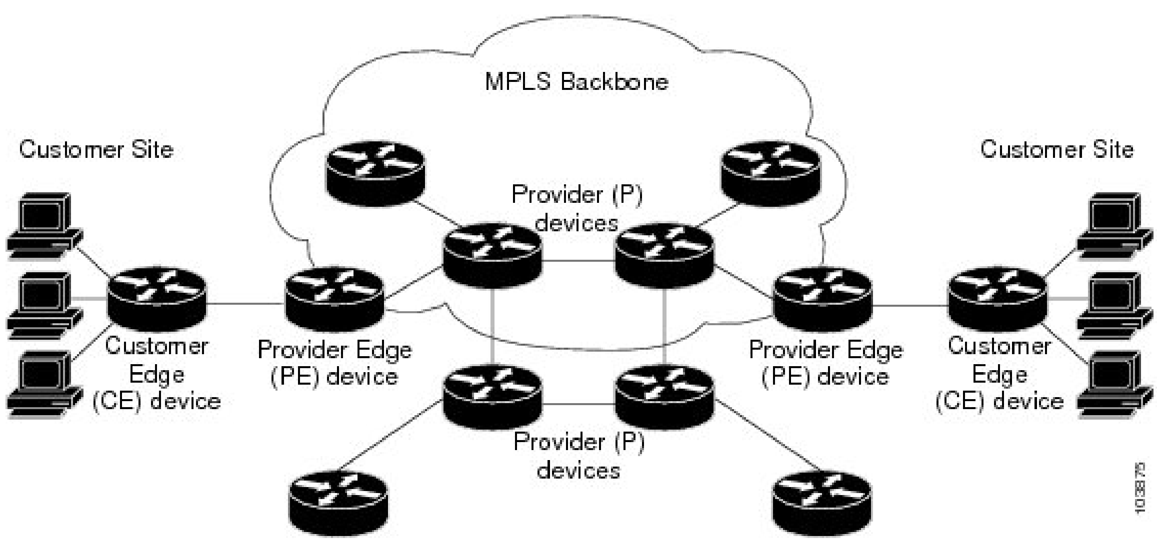
MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the service provider's edge device that provides services to the customer site needs to be updated.

The different parts of the MPLS VPN are described as follows:

- Provider (P) device—Device in the core of the provider network. P devices run MPLS switching, and do not attach VPN labels to routed packets. The MPLS label in each route is assigned by the provider edge (PE) device. VPN labels are used to direct data packets to the correct egress device.
- PE device—Device that attaches the VPN label to incoming packets based on the interface or subinterface on which they are received. A PE device attaches directly to a customer edge (CE) device.
- Customer (C) device—Device in the ISP or enterprise network.
- CE device—Edge device on the network of the ISP that connects to the PE device on the network. A CE device must interface with a PE device.

The figure below shows a basic MPLS VPN.

Figure 6: Basic MPLS VPN Terminology



How an MPLS Virtual Private Network Works

Multiprotocol Label Switching virtual private network (MPLS VPN) functionality is enabled at the edge of an MPLS network. The provider edge (PE) device performs the following:

- Exchanges routing updates with the customer edge (CE) device.
- Translates the CE routing information into VPNv4 routes.
- Exchanges VPNv4 routes with other PE devices through the Multiprotocol Border Gateway Protocol (MP-BGP).

The following sections describe how MPLS VPN works:

Major Components of an MPLS Virtual Private Network

A Multiprotocol Label Switching (MPLS)-based virtual private network (VPN) 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 VPN community provider edge (PE) devices— MP-BGP propagates virtual routing and forwarding (VRF) reachability information to all members of a VPN community. MP-BGP peering must be configured on all PE devices 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.

Benefits of an MPLS Virtual Private Network

Multiprotocol Label Switching virtual private networks (MPLS VPNs) allow service providers to deploy scalable VPNs. They build the foundation to deliver value-added services, such as the following:

Connectionless Service

A significant technical advantage of MPLS VPNs is that they are connectionless. The Internet owes its success to its basic technology, TCP/IP. TCP/IP is built on a packet-based, connectionless network paradigm. This means that no prior action is necessary to establish communication between hosts, making it easy for two parties to communicate. To establish privacy in a connectionless IP environment, current VPN solutions impose a connection-oriented, point-to-point overlay on the network. Even if it runs over a connectionless network, a VPN cannot take advantage of the ease of connectivity and multiple services available in connectionless networks. When you create a connectionless VPN, you do not need tunnels and encryption for network privacy, thus eliminating significant complexity.

Centralized Service

Building VPNs in Layer 3 allows delivery of targeted services to a group of users represented by a VPN. A VPN must give service providers more than a mechanism for privately connecting users to intranet services. It must also provide a way to flexibly deliver value-added services to targeted customers. Scalability is critical, because you want to use services privately in their intranets and extranets. Because MPLS VPNs are seen as private intranets, you may use new IP services such as:

- Multicast
- Quality of service (QoS)
- Telephony support within a VPN
- Centralized services including content and web hosting to a VPN

You can customize several combinations of specialized services for individual customers. For example, a service that combines IP multicast with a low-latency service class enables video conferencing within an intranet.

Scalability

If you create a VPN using connection-oriented, point-to-point overlays, Frame Relay, or ATM virtual connections (VCs), the VPN's key deficiency is scalability. Specifically, connection-oriented VPNs without fully meshed connections between customer sites are not optimal. MPLS-based VPNs, instead, use the peer model and Layer 3 connectionless architecture to leverage a highly scalable VPN solution. The peer model requires a customer site to peer with only one provider edge (PE) device as opposed to all other customer edge (CE) devices that are members of the VPN. The connectionless architecture allows the creation of VPNs in Layer 3, eliminating the need for tunnels or VCs.

Other scalability issues of MPLS VPNs are due to the partitioning of VPN routes between PE devices. And the further partitioning of VPN and Interior Gateway Protocol (IGP) routes between PE devices and provider (P) devices in a core network.

- PE devices must maintain VPN routes for those VPNs who are members.
- P devices do not maintain any VPN routes.

This increases the scalability of the provider's core and ensures that no one device is a scalability bottleneck.

Security

MPLS VPNs offer the same level of security as connection-oriented VPNs. Packets from one VPN do not inadvertently go to another VPN.

Security is provided in the following areas:

- At the edge of a provider network, ensuring packets that are received from a customer are placed on the correct VPN.
- At the backbone, VPN traffic is kept separate. Malicious spoofing (an attempt to gain access to a PE device) is nearly impossible because the packets that are received from customers are IP packets. These IP packets must be received on a particular interface or subinterface to be uniquely identified with a VPN label.

Ease of Creation

To take full advantage of VPNs, customers must be able to easily create new VPNs and user communities. Because MPLS VPNs are connectionless, no specific point-to-point connection maps or topologies are required. You can add sites to intranets and extranets and form closed user groups. Managing VPNs in this manner enables membership of any given site in multiple VPNs, maximizing flexibility in building intranets and extranets.

Flexible Addressing

To make a VPN service more accessible, customers of a service provider can design their own addressing plan. This addressing plan can be independent of addressing plans for other service provider customers. Many customers use private address spaces, as defined in RFC 1918. They do not want to invest the time and expense of converting to public IP addresses to enable intranet connectivity. MPLS VPNs allow customers to continue to use their present address spaces without Network Address Translation (NAT) by providing a public and private view of the address. A NAT is required only if two VPNs with overlapping address spaces want to communicate. This enables customers to use their own unregistered private addresses, and communicate freely across a public IP network.

Integrated QoS Support

QoS is an important requirement for many IP VPN customers. It provides the ability to address two fundamental VPN requirements:

- Predictable performance and policy implementation
- Support for multiple levels of service in an MPLS VPN

Network traffic is classified and labeled at the edge of the network. The traffic is then aggregated according to policies defined by subscribers and implemented by the provider and transported across the provider core. Traffic at the edge and core of the network can then be differentiated into different classes by drop probability or delay.

Straightforward Migration

For service providers to quickly deploy VPN services, use a straightforward migration path. MPLS VPNs are unique because you can build them over multiple network architectures, including IP, ATM, Frame Relay, and hybrid networks.

Migration for the end customer is simplified because there is no requirement to support MPLS on the CE device. No modifications are required to a customer's intranet.

How to Configure MPLS Virtual Private Networks

The following section provides the steps to configure MPLS Virtual Private Networks:

Configuring the Core Network

The following section provides the steps to configure the core network:

Assessing the Needs of MPLS Virtual Private Network Customers

Before you configure a Multiprotocol Label Switching virtual private network (MPLS VPN), you need to identify the core network topology so that it can best serve MPLS VPN customers. Perform this task to identify the core network topology.

Procedure

	Command or Action	Purpose
Step 1	Identify the size of the network.	Identify the following to determine the number of devices and ports that you need:

	Command or Action	Purpose
		• How many customers do you need to support? • How many VPNs are needed per customer? • How many virtual routing and forwarding instances are there for each VPN?
Step 2	Identify the routing protocols in the core.	Determine which routing protocols you need in the core network.
Step 3	Determine if you need MPLS VPN High Availability support.	MPLS VPN Nonstop Forwarding and Graceful Restart are supported on select devices and Cisco software releases. Contact Cisco Support for the exact requirements and hardware support.
Step 4	Determine if you need Border Gateway Protocol (BGP) load sharing and redundant paths in the MPLS VPN core.	For configuration steps, see the "Load Sharing MPLS VPN Traffic" feature module in the *MPLS Layer 3 VPNs Inter-AS and CSC Configuration Guide*.

Configuring MPLS in the Core

To enable Multiprotocol Label Switching (MPLS) on all devices in the core, you must configure either of the following as a label distribution protocol:

- MPLS Label Distribution Protocol (LDP). For configuration information, see the "MPLS Label Distribution Protocol (LDP)" module in the *MPLS Label Distribution Protocol Configuration Guide*.

Connecting the MPLS Virtual Private Network Customers

The following section provides information about Connecting the MPLS Virtual Private Network Customers:

Defining VRFs on the PE Devices to Enable Customer Connectivity

Use this procedure to define a virtual routing and forwarding (VRF) configuration for IPv4. To define a VRF for IPv4 and IPv6, see the "Configuring a Virtual Routing and Forwarding Instance for IPv6"section in the "IPv6 VPN over MPLS" module in the *MPLS Layer 3 VPNs Configuration Guide*.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vrf1`	Defines the virtual private network (VPN) routing instance by assigning a virtual routing and forwarding (VRF) name and enters VRF configuration mode. • The *vrf-name* argument is the name assigned to a VRF.

Step 4	**rd** *route-distinguisher*
	Example:
	`Device(config-vrf)# rd 100:1`
		Creates routing and forwarding tables.
		• The *route-distinguisher* argument adds an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter a route distinguisher (RD) in either of these formats:
		  • 16-bit AS number:your 32-bit number, for example, 101:3
		  • 32-bit IP address:your 16-bit number, for example, 10.0.0.1:1

Step 5	**address-family** *ipv4* | *ipv6* **Example:** `Device(config-vrf)# address-family ipv6`	Enters IPv4 or IPv6 address family mode
Step 6	**route-target** {**import** | **export** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target both 100:1`	Creates a route-target extended community for a VRF. • The **import** keyword imports routing information from the target VPN extended community. • The **export** keyword exports routing information to the target VPN extended community. • The **both** keyword imports routing information from and exports routing information to the target VPN extended community. • The *route-target-ext-community* argument adds the route-target extended community attributes to the VRF's list of import,

	Command or Action	Purpose
		export, or both route-target extended communities.
Step 7	**exit** **Example:** `Device(config-vrf)# exit`	(Optional) Exits to global configuration mode.

Configuring VRF Interfaces on PE Devices for Each VPN Customer

To associate a virtual routing and forwarding (VRF) instance with an interface or subinterface on the provider edge (PE) devices, perform this task.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 0/0/1`	Specifies the interface to configure and enters interface configuration mode. • The *type* argument specifies the type of interface to be configured. • The *number* argument specifies the port, connector, or interface card number.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding vrf1`	Associates a VRF with the specified interface or subinterface. • The *vrf-name* argument is the name that is assigned to a VRF.
Step 5	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.

Configuring Routing Protocols Between the PE and CE Devices

Configure the provider edge (PE) device with the same routing protocol that the customer edge (CE) device uses. You can configure the Border Gateway Protocol (BGP), Routing Information Protocol version 2 (RIPv2), EIGRP, Open Shortest Path First (OSPF) or static routes between the PE and CE devices.

Verifying the Virtual Private Network Configuration

A route distinguisher must be configured for the virtual routing and forwarding (VRF) instance. Multiprotocol Label Switching (MPLS) must be configured on the interfaces that carry the VRF. Use the **show ip vrf** command to verify the route distinguisher (RD) and interface configured for the VRF.

Procedure

show ip vrf

Displays the set of defined VRF instances and associated interfaces. The output also maps the VRF instances to the configured route distinguisher.

Verifying Connectivity Between MPLS Virtual Private Network Sites

To verify that the local and remote customer edge (CE) devices can communicate across the Multiprotocol Label Switching (MPLS) core, perform the following tasks:

Verifying IP Connectivity from CE Device to CE Device Across the MPLS Core

Procedure

Step 1 **enable**

Enables privileged EXEC mode.

Step 2 **ping** [*protocol*] {*host-name* | *system-address*}

Diagnoses basic network connectivity on AppleTalk, Connectionless-mode Network Service (CLNS), IP, Novell, Apollo, Virtual Integrated Network Service (VINES), DECnet, or Xerox Network Service (XNS) networks. Use the **ping** command to verify the connectivity from one CE device to another.

Step 3 **trace** [*protocol*] [*destination*]

Discovers the routes that packets take when traveling to their destination. The **trace** command can help isolate a trouble spot if two devices cannot communicate.

Step 4 **show ip route** [*ip-address* [*mask*] [**longer-prefixes**]] | *protocol* [*process-id*]] | [**list** [*access-list-name* | *access-list-number*]

Displays the current state of the routing table. Use the *ip-address* argument to verify that CE1 has a route to CE2. Verify the routes learned by CE1. Make sure that the route for CE2 is listed.

Verifying That the Local and Remote CE Devices Are in the PE Routing Table

Procedure

Step 1 **enable**

Enables privileged EXEC mode.

Step 2 **show ip route vrf** *vrf-name* [*prefix*]

Displays the IP routing table that is associated with a virtual routing and forwarding (VRF) instance. Check that the loopback addresses of the local and remote customer edge (CE) devices are in the routing table of the provider edge (PE) devices.

Step 3 **show ip cef vrf** *vrf-name* [*ip-prefix*]

Displays the Cisco Express Forwarding forwarding table that is associated with a VRF. Check that the prefix of the remote CE device is in the Cisco Express Forwarding table.

Configuration Examples for MPLS Virtual Private Networks

The following section provides the configuration examples for MPLS Virtual Private Networks:

Example: Configuring an MPLS Virtual Private Network Using RIP

<table>
<tr><th>PE Configuration</th><th>CE Configuration</th></tr>
<tr><td>

```
 vrf vpn1
 rd 100:1
 route-target export 100:1
 route-target import 100:1
!
ip cef
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
interface Loopback0
 ip address 10.0.0.1 255.255.255.255
!
interface GigabitEthernet 1/0/1
 vrf forwarding vpn1
 ip address 192.0.2.3 255.255.255.0
 no cdp enable
interface GigabitEthernet 1/0/1
ip address 192.0.2.2 255.255.255.0
mpls label protocol ldp
mpls ip
!
router rip
version 2
timers basic 30 60 60 120
!
address-family ipv4 vrf vpn1
version 2
redistribute bgp 100 metric transparent
network 192.0.2.0
distribute-list 20 in
no auto-summary
exit-address-family
!
router bgp 100
no synchronization
bgp log-neighbor changes
neighbor 10.0.0.3 remote-as 100
neighbor 10.0.0.3 update-source Loopback0
no auto-summary
!
address-family vpnv4
 neighbor 10.0.0.3 activate
 neighbor 10.0.0.3 send-community extended

 bgp scan-time import 5
 exit-address-family
!
address-family ipv4 vrf vpn1
 redistribute connected
 redistribute rip
 no auto-summary
 no synchronization
 exit-address-family
```

</td><td>

```
ip cef
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
interface Loopback0
 ip address 10.0.0.9 255.255.255.255
!
interface GigabitEthernet 1/0/1
 ip address 192.0.2.1 255.255.255.0
 no cdp enable
router rip
 version 2
 timers basic 30 60 60 120
 redistribute connected
 network 10.0.0.0
 network 192.0.2.0
 no auto-summary
```

</td></tr>
</table>

Example: Configuring an MPLS Virtual Private Network Using Static Routes

<table>
<tr><th>PE Configuration</th><th>CE Configuration</th></tr>
<tr><td><pre>vrf vpn1
 rd 100:1
 route-target export 100:1
 route-target import 100:1
!
ip cef
mpls ldp router-id Loopback0 force
mpls label protocol ldp
!
interface Loopback0
 ip address 10.0.0.1 255.255.255.255
!
interface GigabitEthernet 1/0/1
 vrf forwarding vpn1
 ip address 192.0.2.3 255.255.255.0
 no cdp enable
!
interface GigabitEthernet 1/0/1
ip address 192.168.0.1 255.255.0.0
mpls label protocol ldp
mpls ip
!
router ospf 100
network 10.0.0. 0.0.0.0 area 100
network 192.168.0.0 255.255.0.0 area 100
!
router bgp 100
 no synchronization
 bgp log-neighbor changes
 neighbor 10.0.0.3 remote-as 100
 neighbor 10.0.0.3 update-source Loopback0
no auto-summary
 !
address-family vpnv4
 neighbor 10.0.0.3 activate
 neighbor 10.0.0.3 send-community extended
 bgp scan-time import 5
 exit-address-family
 !
address-family ipv4 vrf vpn1
 redistribute connected
 redistribute static
 no auto-summary
 no synchronization
 exit-address-family
!
ip route vrf vpn1 10.0.0.9 255.255.255.255
192.0.2.2
ip route vrf vpn1 192.0.2.0 255.255.0.0
192.0.2.2</pre></td><td><pre>ip cef
!
interface Loopback0
 ip address 10.0.0.9 255.255.255.255
!
interface GigabitEthernet 1/0/1
 ip address 192.0.2.2 255.255.0.0
 no cdp enable
!
ip route 10.0.0.9 255.255.255.255 192.0.2.3
3
ip route 198.51.100.0 255.255.255.0 192.0.2.3
 3</pre></td></tr>
</table>

Example: Configuring an MPLS Virtual Private Network Using BGP

PE Configuration	CE Configuration

	router bgp 5000
	bgp log-neighbor-changes
	neighbor 5.5.5.6 remote-as 5001
	neighbor 5.5.5.6 ebgp-multihop 2
	neighbor 5.5.5.6 update-source Loopback5
	neighbor 35.2.2.2 remote-as 5001
	neighbor 35.2.2.2 ebgp-multihop 2
	neighbor 35.2.2.2 update-source Loopback1
	neighbor 3500::1 remote-as 5001
	neighbor 3500::1 ebgp-multihop 2
	neighbor 3500::1 update-source Loopback1
	!
	address-family ipv4
	  redistribute connected
	  neighbor 5.5.5.6 activate
	  neighbor 35.2.2.2 activate
	  no neighbor 3500::1 activate
	exit-address-family
	!
	address-family ipv6
	  redistribute connected
	  neighbor 3500::1 activate
	exit-address-family
	Device-RP(config)#

PE Configuration	CE Configuration

router bgp 5001
bgp log-neighbor-changes
bgp graceful-restart
bgp sso route-refresh-enable
bgp refresh max-eor-time 600
redistribute connected
neighbor 102.1.1.1 remote-as 5001
neighbor 102.1.1.1 update-source Loopback1
neighbor 105.1.1.1 remote-as 5001
neighbor 105.1.1.1 update-source Loopback10
neighbor 160.1.1.2 remote-as 5002
!
address-family vpnv4
  neighbor 102.1.1.1 activate
  neighbor 102.1.1.1 send-community both
  neighbor 105.1.1.1 activate
  neighbor 105.1.1.1 send-community extended
exit-address-family
!
address-family vpnv6
  neighbor 102.1.1.1 activate
  neighbor 102.1.1.1 send-community extended
  neighbor 105.1.1.1 activate
  neighbor 105.1.1.1 send-community extended
exit-address-family
!
address-family ipv4 vrf full
  redistribute connected
  neighbor 20.1.1.1 remote-as 5000
  neighbor 20.1.1.1 ebgp-multihop 2
  neighbor 20.1.1.1 update-source Loopback2
  neighbor 20.1.1.1 activate
  neighbor 20.1.1.1 send-community both
exit-address-family
!
address-family ipv6 vrf full
  redistribute connected
  neighbor 2000::1 remote-as 5000
  neighbor 2000::1 ebgp-multihop 2
  neighbor 2000::1 update-source Loopback2
  neighbor 2000::1 activate
exit-address-family
!
address-family ipv4 vrf orange
  network 87.1.0.0 mask 255.255.252.0
  network 87.1.1.0 mask 255.255.255.0
  redistribute connected
  neighbor 40.1.1.1 remote-as 7000
  neighbor 40.1.1.1 ebgp-multihop 2
  neighbor 40.1.1.1 update-source Loopback3
  neighbor 40.1.1.1 activate
  neighbor 40.1.1.1 send-community extended
  neighbor 40.1.1.1 route-map orange-lp in
  maximum-paths eibgp 2
exit-address-family
!
address-family ipv6 vrf orange
  redistribute connected
  maximum-paths eibgp 2
  neighbor 4000::1 remote-as 7000
  neighbor 4000::1 ebgp-multihop 2
  neighbor 4000::1 update-source Loopback3	

<table>
<tr><th>PE Configuration</th><th>CE Configuration</th></tr>
<tr><td><pre>
  neighbor 4000::1 activate
exit-address-family
!
address-family ipv4 vrf sona
  redistribute connected
  neighbor 160.1.1.2 remote-as 5002
  neighbor 160.1.1.2 activate
  neighbor 160.1.1.4 remote-as 5003
  neighbor 160.1.1.4 activate
exit-address-family
</pre></td><td></td></tr>
</table>

Additional References

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the MPLS Commands section of the *Command Reference (Catalyst 9500 Series Switches)*
Configuring Cisco Express Forwarding	"Configuring Basic Cisco Express Forwarding" module in the *Cisco Express Forwarding Configuration Guide*
Configuring LDP	"MPLS Label Distribution Protocol (LDP)" module in the *MPLS Label Distribution Protocol Configuration Guide*

Feature History for MPLS Virtual Private Networks

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	MPLS Virtual Private Networks	An MPLS Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of a Multiprotocol Label Switching (MPLS) provider core network. At each customer site, one or more customer edge (CE) devices attach to one or more provider edge (PE) devices.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 8

Configuring MPLS InterAS Option B

Information About MPLS VPN InterAS Options

The MPLS VPN InterAS Options provide various ways of interconnecting VPNs between different MPLS VPN service providers. This allows sites of a customer to exist on several carrier networks (autonomous systems) and have seamless VPN connectivity between these sites.

ASes and ASBRs

An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and using a single, clearly defined protocol. In many cases, VPNs extend to different ASes in different geographical areas. Some VPNs must extend across multiple service providers; these VPNs are called overlapping VPNs. The connection between ASes must be seamless to the customer, regardless of the complexity or location of the VPNs.

An AS boundary router (ASBR) is a device in an AS that is connected by using more than one routing protocol, and exchanges routing information with other ASBRs by using an exterior routing protocol (for example, eBGP), or use static routes, or both.

Separate ASes from different service providers communicate by exchanging information in the form of VPN IP addresses and they use the following protocols to share routing information:

- Within an AS, routing information is shared using iBGP.

  iBGP distributes network layer information for IP prefixes within each VPN and each AS.

- Between ASes, routing information is shared using eBGP.

  eBGP allows service providers to set up an interdomain routing system that guarantees loop-free exchange of routing information between separate ASes. The primary function of eBGP is to exchange network reachability information between ASes, including information about the list of AS routes. The ASes use eBGP border edge routers to distribute the routes, which includes label-switching information. Each border edge router rewrites the next-hop and MPLS labels.

MPLS VPN InterAS Options configuration is supported and can include an inter provider VPN, which is MPLS VPNs that include two or more ASes, connected by separate border edge routers. The ASes exchange routes using eBGP, and no iBGP or routing information is exchanged between the ASes.

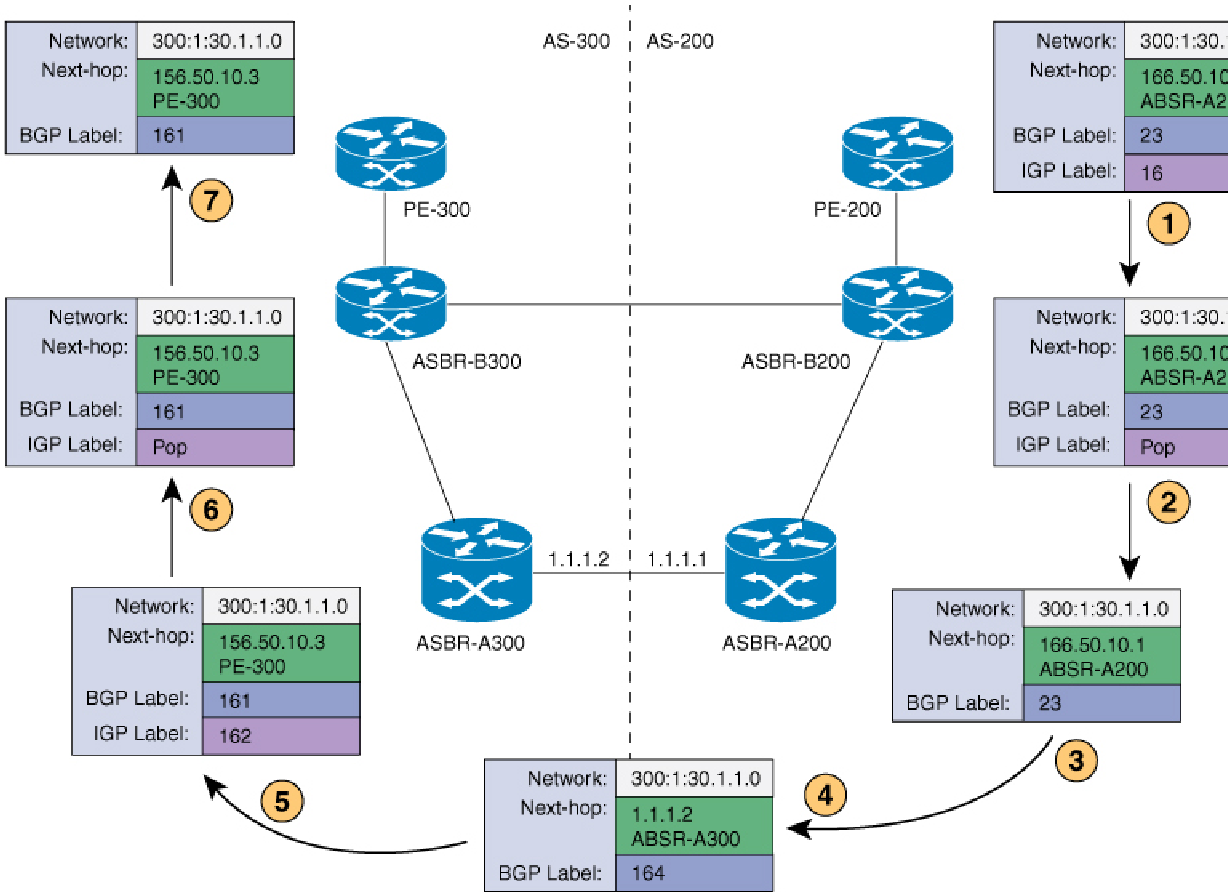
MPLS VPN InterAS Options

The following options defined in RFC4364 provide MPLS VPN connectivity between different ASes:

- InterAS Option A – This option provides back-to-back virtual routing and forwarding (VRF) connectivity. Here, MPLS VPN providers exchange routes across VRF interfaces.
- InterAS Option B – This option provides VPNv4 route distribution between ASBRs.

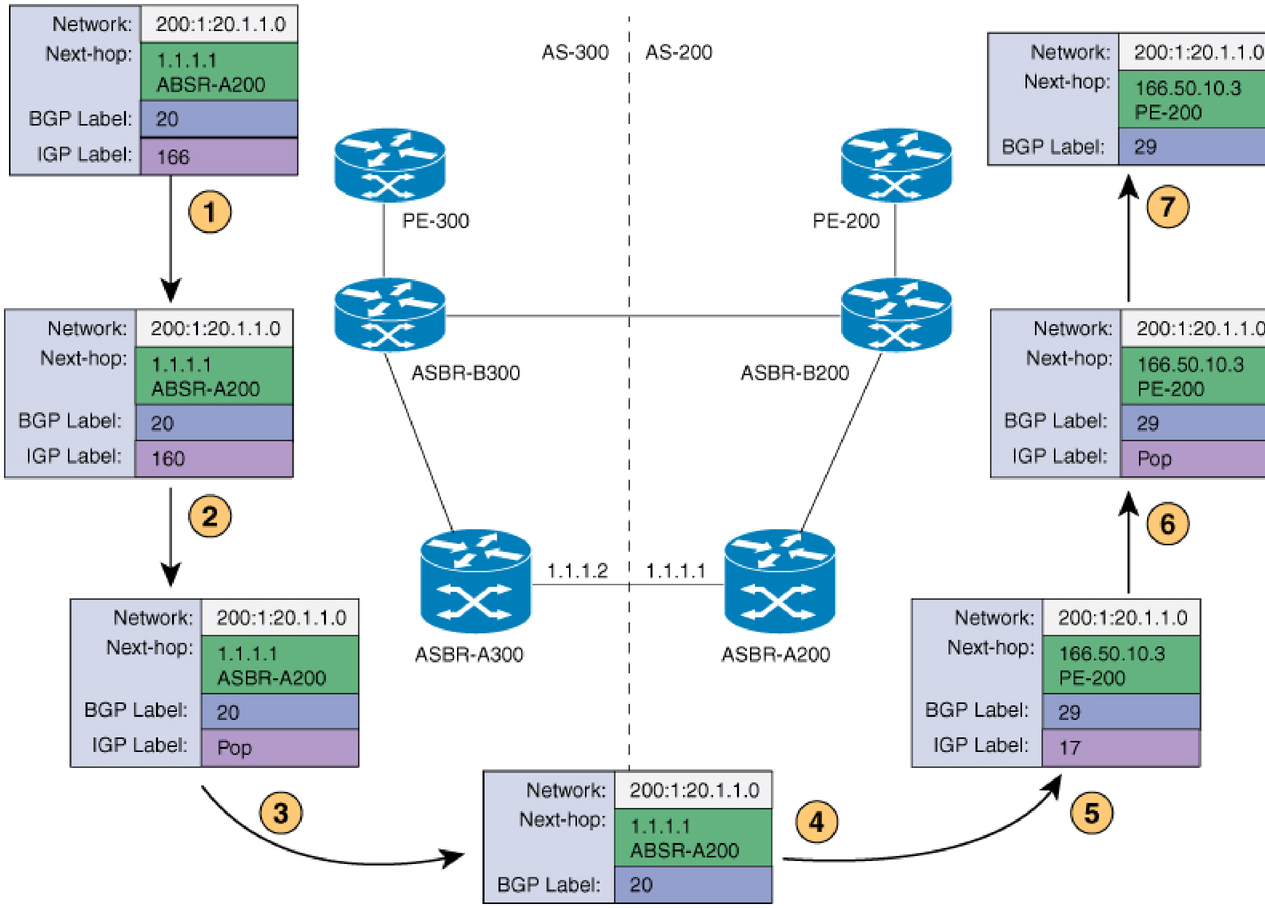
Next-Hop Self Method

The following figure shows the label forwarding path for next-hop-self method. The labels get pushed, swapped and popped on the stack as packet makes its way from PE-200 in AS 200 to PE-300 in AS 300. In step 5, ASBR-A300 receives labeled frame, replaces label 164 with label 161 pushes IGP label 162 onto the label stack.



Redistribute Connected Subnet Method

The following figure shows the label forwarding path for Redistribute connected subnets method. The labels get pushed, swapped and popped on the stack as packet travels from PE- 300 in AS 300 to PE-200 in AS 200. In step 5, ASBR-A200 receives frame with BGP label 20, swaps it with label 29 and pushes label 17.



Configuring MPLS VPN InterAS Option B

Configuring InterAS Option B using the Next-Hop-Self Method

To configure interAS Option B on ASBRs using the next-hop-self method, complete the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**router ospf** *process-id* **Example:** `Device(config)# router ospf 1`	Configures an OSPF routing process and assign a process number.
Step 4	**router-id** *ip-address* **Example:** `Device(config)# router-id 4.1.1.1`	Specifies a fixed router ID.
Step 5	**nsr** **Example:** `Device(config-router)# nsr`	Configures OSPF non-stop routing (NSR).
Step 6	**nsf** **Example:** `Device(config-router)# nsf`	Confgures OSPF non-stop forwarding (NSF).
Step 7	**redistribute bgp** *autonomous-system-number* **Example:** `Device(config-router)# redistribute bgp 200`	Redistributes routes from a BGP autonomous system into and OSPF routing process.

Step 8	**passive-interface** *interface-type interface-number*
	Example:
	`Device(config-router)# passive-interface GigabitEthernet 1/0/10`
	`Device(config-router)# passive-interface Tunnel0`
		Disables Open Shortest Path First (OSPF) routing updates on an interface.

Step 9	**network** *ip-address wildcard-mask* **aread** *area-id* **Example:** `Device(config-router)# network 4.1.1.0 0.0.0.0.255 area 0`	Defines an interface on which OSPF runs and defines the area ID for that interface.
Step 10	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode.

	Command or Action	Purpose
Step 11	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 200`	Configures a BGP routing process.
Step 12	**bgp router-id** *ip-address* **Example:** `Device(config-router)# bgp router-id 4.1.1.1`	Configures a fixed router ID for the BGP routing process.
Step 13	**bgp log-neighbor changes** **Example:** `Device(config-router)# bgp log-neighbor changes`	Enables logging of BGP neighbor resets.
Step 14	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	Disables advertisement of routing information for address family IPv4.
Step 15	**no bgp default route-target filter** **Example:** `Device(config-router)# no bgp default route-target filter`	Disables automatic BGP route-target community filtering.
Step 16	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 4.1.1.3 remote-as 200`	Configures an entry to the BGP neighbor table.
Step 17	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 4.1.1.3 update-source Loopback0`	Allows Cisco IOS software to use a specific operational interface for TCP connections by the BGP sessions.
Step 18	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 4.1.1.3 remote-as 300`	Configures an entry to the BGP neighbor table.

	Command or Action	Purpose
Step 19	**address-family** *ipv4* **Example:** `Device(config-router)# address-family ipv4`	Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.
Step 20	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.32.1.2 activate`	Enables the exchange of information with a BGP neighbor.
Step 21	**neighbor** *ip-address* **send-label** **Example:** `Device(config-router-af)# neighbor 10.32.1.2 send-label`	Sends MPLS labels with BGP routes to a neighboring BGP router.
Step 22	**exit address-family** **Example:** `Device(config-router-af)# exit address-family`	Exits BGP address-family submode.
Step 23	**address-family** *vpnv4* **Example:** `Device(config-router)# address-family vpnv4`	Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.
Step 24	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 4.1.1.3 activate`	Enables the exchange of information with a BGP neighbor.
Step 25	**neighbor** *ip-address* **send-community extended** **Example:** `Device(config-router-af)# neighbor 4.1.1.3 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 26	**neighbor** *ip-address* **next-hop-self** **Example:** `Device(config-router-af)# neighbor 4.1.1.3 next-hop-self`	Configure a router as the next hop for a BGP-speaking neighbor. This is the command that implements the next-hop-self method.

	Command or Action	Purpose
Step 27	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.30.1.2 activate`	Enables the exchange of information with a BGP neighbor.
Step 28	**neighbor** *ip-address* **send-community extended** **Example:** `Device(config-router-af)# neighbor 10.30.1.2 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 29	**exit address-family** **Example:** `Device(config-router-af)# exit address-family`	Exits BGP address-family submode.
Step 30	**bgp router-id** *ip-address* **Example:** `Device(config-router)# bgp router-id 4.1.1.3`	Configures a fixed router ID for the BGP routing process.
Step 31	**bgp log-neighbor changes** **Example:** `Device(config-router)# bgp log-neighbor changes`	Enables logging of BGP neighbor resets.
Step 32	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 4.1.1.1 remote-as 200`	Configures an entry to the BGP neighbor table.
Step 33	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 4.1.1.1 update-source Loopback0`	Allows Cisco IOS software to use a specific operational interface for TCP connections by the BGP sessions.
Step 34	**address-family** *vpnv4* **Example:** `Device(config-router)# address-family vpnv4`	Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.

	Command or Action	Purpose
Step 35	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 4.1.1.1 activate`	Enables the exchange of information with a BGP neighbor.
Step 36	**neighbor** *ip-address* **send-community extended** **Example:** `Device(config-router-af)# neighbor 4.1.1.1 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 37	**exit address-family** **Example:** `Device(config-router-af)# exit address-family`	Exits BGP address-family submode.

Configuring InterAS Option B using Redistribute Connected Method

To configure interAS Option B on ASBRs using the redistribute connected method, complete the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **Example:** `Device(config)# router ospf 1`	Configures an OSPF routing process and assign a process number.
Step 4	**router-id** *ip-address* **Example:**	Specifies a fixed router ID.

	Command or Action	Purpose
	`Device(config)# router-id 5.1.1.1`	
Step 5	**nsr** **Example:** `Device(config-router)# nsr`	Configures OSPF non-stop routing (NSR).
Step 6	**nsf** **Example:** `Device(config-router)# nsf`	Confgures OSPF non-stop forwarding (NSF).
Step 7	**redistribute connected** **Example:** `Device(config-router)# redistribute connected`	Redistributes the next hop address of the remote ASBR into the local IGP. This is the command that implements redistribute connected method.

Step 8	**passive-interface** *interface-type interface-number*
	Example:
	`Device(config-router)# passive-interface GigabitEthernet 1/0/10`
	`Device(config-router)# passive-interface Tunnel0`
		Disables Open Shortest Path First (OSPF) routing updates on an interface.

Step 9	**network** *ip-address wildcard-mask* **aread** *area-id* **Example:** `Device(config-router)# network 5.1.1.0 0.0.0.0.255 area 0`	Defines an interface on which OSPF runs and defines the area ID for that interface.
Step 10	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode.
Step 11	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 300`	Configures a BGP routing process.
Step 12	**bgp router-id** *ip-address* **Example:** `Device(config-router)# bgp router-id 5.1.1.1`	Configures a fixed router ID for the BGP routing process.

	Command or Action	Purpose
Step 13	**bgp log-neighbor changes** **Example:** `Device(config-router)# bgp log-neighbor changes`	Enables logging of BGP neighbor resets.
Step 14	**no bgp default ipv4-unicast** **Example:** `Device(config-router)# no bgp default ipv4-unicast`	Disables advertisement of routing information for address family IPv4.
Step 15	**no bgp default route-target filter** **Example:** `Device(config-router)# no bgp default route-target filter`	Disables automatic BGP route-target community filtering.
Step 16	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 5.1.1.3 remote-as 300`	Configures an entry to the BGP neighbor table.
Step 17	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 4.1.1.3 update-source Loopback0`	Allows Cisco IOS software to use a specific operational interface for TCP connections by the BGP sessions.
Step 18	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 10.30.1.2 remote-as 200`	Configures an entry to the BGP neighbor table.
Step 19	**address-family** *vpnv4* **Example:** `Device(config-router)# address-family vpnv4`	Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.
Step 20	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 5.1.1.3 activate`	Enables the exchange of information with a BGP neighbor.

	Command or Action	Purpose
Step 21	**neighbor** *ip-address* **send-community extended** **Example:** `Device(config-router-af)# neighbor 5.1.1.3 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 22	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.30.1.1 activate`	Enables the exchange of information with a BGP neighbor.
Step 23	**neighbor** *ip-address* **send-community extended** **Example:** `Device(config-router-af)# neighbor 10.30.1.2 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 24	**exit address-family** **Example:** `Device(config-router-af)# exit address-family`	Exits BGP address-family submode.
Step 25	**mpls ldp router-id** *interface-id* [**force**] **Example:** `Device(config-router)# mpls ldp router-id Loopback0 force`	Specifies the preferred interface for determining the LDP router ID.

Verifying MPLS VPN InterAS Options Configuration

To verify InterAS option B configuration information, perform one of the following tasks:

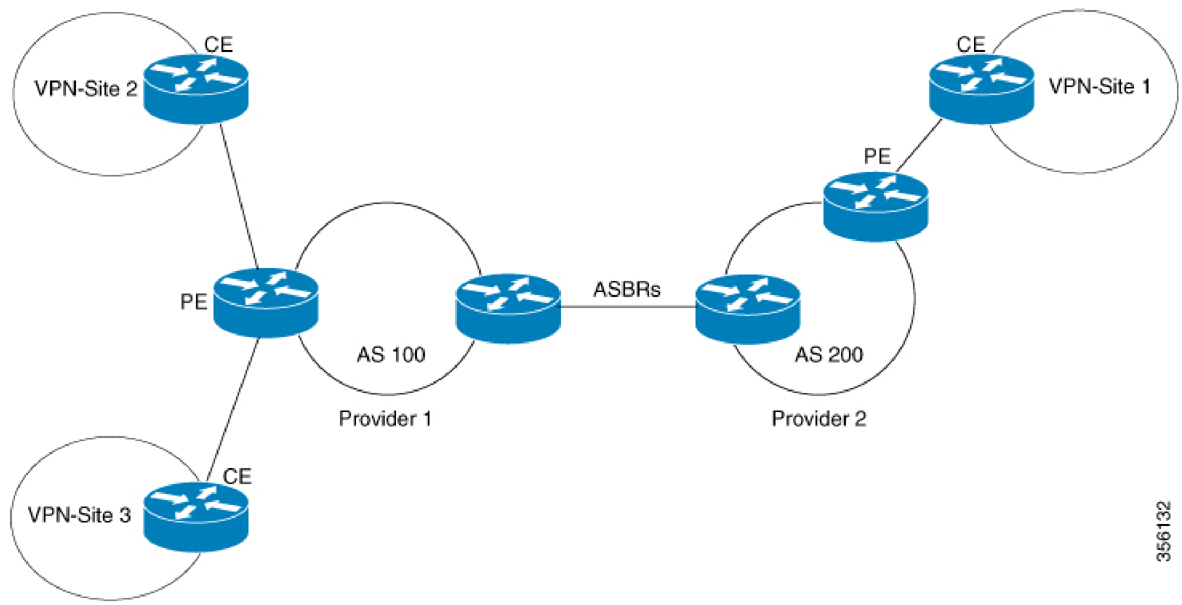
Command	Purpose
ping *ip-address* **source** *interface-type*	Checks the accessibility of devices. Use this command to check the connection between CE1 and CE2 using the loopback interface.
show bgp vpnv4 unicast labels	Displays incoming and outgoing BGP labels.
show mpls forwarding-table	Display the contents of the MPLS Label Forwarding Information Base.
show ip bgp	Displays entries in the BGP routing table.

Command	Purpose
show { **ip** | **ipv6** } **bgp** [**vrf** *vrf-name*]	Displays information about BGP on a VRF.
show ip route [*ip-address* [*mask*]] [*protocol*] **vrf** *vrf-name*	Displays the current state of the routing table. Use the ip-address argument to verify that CE1 has a route to CE2. Verify the routes learned by CE1. Make sure that the route for CE2 is listed.
show { **ip** | **ipv6** } **route vrf** *vrf-name*	Displays the IP routing table that is associated with a VRF. Check that the loopback addresses of the local and remote CE routers are in the routing table of the PE routers.
show running-config bgp	Displays the running configuration for BGP.
show running-config vrf *vrf-name*	Displays the running configuration for VRFs.
show vrf *vrf-name* **interface** *interface-type interface-id*	Verifies the route distinguisher (RD) and interface that are configured for the VRF.
trace destination [**vrf** *vrf-name*]	Discovers the routes that packets take when traveling to their destination. The **trace** command can help isolate a problem if two routers cannot communicate.

Configuration Examples for MPLS VPN InterAS Options

Next-Hop-Self Method

Figure 7: Topology for InterAS Option B using Next-Hop-Self Method



Configuration for PE1-P1-ASBR1

PE1	P1	ASBR1
	interface Loopback0 ip address 4.1.1.2 255.255.255.255 ip ospf 1 area 0 interface GigabitEthernet1/0/4 no switchport ip address 10.10.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp ! interface GigabitEthernet1/0/23 no switchport ip address 10.20.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp	interface Loopback0 ip address 4.1.1.1 255.255.255.255 ip ospf 1 area 0 interface GigabitEthernet1/0/10 no switchport ip address 10.30.1.1 255.255.255.0 mpls bgp forwarding interface GigabitEthernet1/0/23 no switchport ip address 10.20.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp router ospf 1 router-id 4.1.1.1 nsr nsf redistribute bgp 200 passive-interface GigabitEthernet1/0/10 passive-interface Tunnel0 network 4.1.1.0 0.0.0.255 area 0 router bgp 200 bgp router-id 4.1.1.1 bgp log-neighbor-changes no bgp default ipv4-unicast no bgp default route-target filter neighbor 4.1.1.3 remote-as 200 neighbor 4.1.1.3 update-source Loopback0 neighbor 10.30.1.2 remote-as 300 ! address-family ipv4 neighbor 10.30.1.2 activate neighbor 10.30.1.2 send-label exit-address-family ! address-family vpnv4 neighbor 4.1.1.3 activate neighbor 4.1.1.3 send-community extended neighbor 4.1.1.3 next-hop-self neighbor 10.30.1.2 activate neighbor 10.30.1.2 send-community extended exit-address-family

PE1	P1	ASBR1
vrf definition Mgmt-vrf ! address-family ipv4 exit-address-family ! address-family ipv6 exit-address-family ! vrf definition vrf1 rd 200:1 route-target export 200:1 route-target import 200:1 route-target import 300:1 ! address-family ipv4 exit-address-family interface Loopback0 ip address 4.1.1.3 255.255.255.255 ip ospf 1 area 0 ! interface Loopback1 vrf forwarding vrf1 ip address 192.1.1.1 255.255.255.255 ip ospf 200 area 0 ! interface GigabitEthernet2/0/4 no switchport ip address 10.10.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp interface GigabitEthernet2/0/9 description to-IXIA-1:p8 no switchport vrf forwarding vrf1 ip address 192.2.1.1 255.255.255.0 ip ospf 200 area 0 router ospf 200 vrf vrf1 router-id 192.1.1.1 nsr nsf redistribute connected redistribute bgp 200 network 192.1.1.1 0.0.0.0 area 0 network 192.2.1.0 0.0.0.255 area 0 router ospf 1 router-id 4.1.1.3 nsr nsf redistribute connected router bgp 200 bgp router-id 4.1.1.3 bgp log-neighbor-changes neighbor 4.1.1.1 remote-as 200 neighbor 4.1.1.1 update-source Loopback0		

PE1	P1	ASBR1
! address-family vpnv4 neighbor 4.1.1.1 activate neighbor 4.1.1.1 send-community extended exit-address-family ! address-family ipv4 vrf vrf1 redistribute connected redistribute ospf 200 maximum-paths ibgp 4 exit-address-family		

Configuration for ASBR2 – P2 – PE2

Table 2:

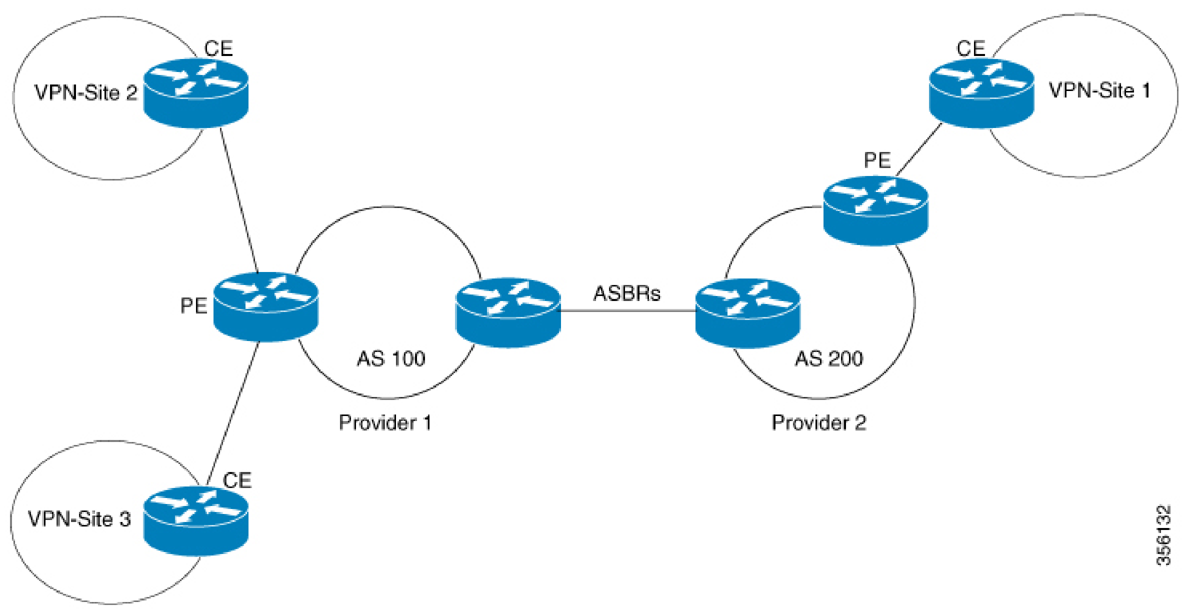
PE2	P2	ASBR2
	interface Loopback0 ip address 5.1.1.2 255.255.255.255 ip ospf 1 area 0 interface GigabitEthernet1/0/1 no switchport ip address 10.50.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp interface GigabitEthernet2/0/3 no switchport ip address 10.40.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp	interface Loopback0 ip address 5.1.1.1 255.255.255.255 ip ospf 1 area 0 ! interface GigabitEthernet1/0/37 no switchport ip address 10.30.1.2 255.255.255.0 mpls bgp forwarding interface GigabitEthernet1/0/47 no switchport ip address 10.40.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp router ospf 1 router-id 5.1.1.1 nsr nsf passive-interface GigabitEthernet1/0/37 passive-interface Tunnel0 network 5.1.1.0 0.0.0.255 area 0 ! router bgp 300 bgp router-id 5.1.1.1 bgp log-neighbor-changes no bgp default ipv4-unicast no bgp default route-target filter neighbor 5.1.1.3 remote-as 300 neighbor 5.1.1.3 update-source Loopback0 neighbor 10.30.1.1 remote-as 200 ! address-family ipv4 neighbor 10.30.1.1 activate neighbor 10.30.1.1 send-label exit-address-family ! address-family vpnv4 neighbor 5.1.1.3 activate neighbor 5.1.1.3 send-community extended neighbor 5.1.1.3 next-hop-self neighbor 10.30.1.1 activate neighbor 10.30.1.1 send-community extended exit-address-family

PE2	P2	ASBR2
vrf definition vrf1 rd 300:1 route-target export 300:1 route-target import 300:1 route-target import 200:1 ! address-family ipv4 exit-address-family interface Loopback0 ip address 5.1.1.3 255.255.255.255 ip ospf 1 area 0 ! interface Loopback1 vrf forwarding vrf1 ip address 193.1.1.1 255.255.255.255 ip ospf 300 area 0 interface GigabitEthernet1/0/1 no switchport ip address 10.50.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp ! interface GigabitEthernet1/0/2 no switchport vrf forwarding vrf1 ip address 193.2.1.1 255.255.255.0 ip ospf 300 area 0 router ospf 300 vrf vrf1 router-id 193.1.1.1 nsr nsf redistribute connected redistribute bgp 300 network 193.1.1.1 0.0.0.0 area 0 network 193.2.1.0 0.0.0.255 area 0 ! router ospf 1 router-id 5.1.1.3 nsr nsf redistribute connected router bgp 300 bgp router-id 5.1.1.3 bgp log-neighbor-changes neighbor 5.1.1.1 remote-as 300 neighbor 5.1.1.1 update-source Loopback0 ! address-family ipv4 neighbor 5.1.1.1 activate neighbor 5.1.1.1 send-label exit-address-family ! address-family vpnv4 neighbor 5.1.1.1 activate		

PE2	P2	ASBR2
neighbor 5.1.1.1 send-community extended exit-address-family ! address-family ipv4 vrf vrf1 redistribute connected redistribute ospf 300 maximum-paths ibgp 4 exit-address-family		

IGP Redistribute Connected Subnets Method

Figure 8: Topology for InterAS Option B using Redistribute Connected Subnets Method



Configuration for PE1-P1-ASBR1

PE1	P1	ASBR1
	interface Loopback0 ip address 4.1.1.2 255.255.255.255 ip ospf 1 area 0 interface GigabitEthernet1/0/4 no switchport ip address 10.10.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp ! interface GigabitEthernet1/0/23 no switchport ip address 10.20.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp	router ospf 1 router-id 4.1.1.1 nsr nsf redistribute connected passive-interface GigabitEthernet1/0/10 passive-interface Tunnel0 network 4.1.1.0 0.0.0.255 area 0 router bgp 200 bgp router-id 4.1.1.1 bgp log-neighbor-changes no bgp default ipv4-unicast no bgp default route-target filter neighbor 4.1.1.3 remote-as 200 neighbor 4.1.1.3 update-source Loopback0 neighbor 10.30.1.2 remote-as 300 ! address-family vpnv4 neighbor 4.1.1.3 activate neighbor 4.1.1.3 send-community extended neighbor 10.30.1.2 activate neighbor 10.30.1.2 send-community extended exit-address-family mpls ldp router-id Loopback0 force

PE1	P1	ASBR1
vrf definition Mgmt-vrf ! address-family ipv4 exit-address-family ! address-family ipv6 exit-address-family ! vrf definition vrf1 rd 200:1 route-target export 200:1 route-target import 200:1 route-target import 300:1 ! address-family ipv4 exit-address-family interface Loopback0 ip address 4.1.1.3 255.255.255.255 ip ospf 1 area 0 ! interface Loopback1 vrf forwarding vrf1 ip address 192.1.1.1 255.255.255.255 ip ospf 200 area 0 ! interface GigabitEthernet2/0/4 no switchport ip address 10.10.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp interface GigabitEthernet2/0/9 description to-IXIA-1:p8 no switchport vrf forwarding vrf1 ip address 192.2.1.1 255.255.255.0 ip ospf 200 area 0 router ospf 200 vrf vrf1 router-id 192.1.1.1 nsr nsf redistribute connected redistribute bgp 200 network 192.1.1.1 0.0.0.0 area 0 network 192.2.1.0 0.0.0.255 area 0 router ospf 1 router-id 4.1.1.3 nsr nsf redistribute connected router bgp 200 bgp router-id 4.1.1.3 bgp log-neighbor-changes neighbor 4.1.1.1 remote-as 200 neighbor 4.1.1.1 update-source Loopback0		

PE1	P1	ASBR1
! address-family vpnv4 neighbor 4.1.1.1 activate neighbor 4.1.1.1 send-community extended exit-address-family ! address-family ipv4 vrf vrf1 redistribute connected redistribute ospf 200 maximum-paths ibgp 4 exit-address-family		

Configuration for ASBR2 – P2 – PE2

PE2	P2	ASBR2
	interface Loopback0 ip address 5.1.1.2 255.255.255.255 ip ospf 1 area 0 interface GigabitEthernet1/0/1 no switchport ip address 10.50.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp interface GigabitEthernet2/0/3 no switchport ip address 10.40.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp	router ospf 1 router-id 5.1.1.1 nsr nsf redistribute connected passive-interface GigabitEthernet1/0/10 passive-interface Tunnel0 network 5.1.1.0 0.0.0.255 area 0 router bgp 300 bgp router-id 5.1.1.1 bgp log-neighbor-changes no bgp default ipv4-unicast no bgp default route-target filter neighbor 5.1.1.3 remote-as 300 neighbor 5.1.1.3 update-source Loopback0 neighbor 10.30.1.1 remote-as 200 ! address-family vpnv4 neighbor 5.1.1.3 activate neighbor 5.1.1.3 send-community extended neighbor 10.30.1.1 activate neighbor 10.30.1.1 send-community extended exit-address-family mpls ldp router-id Loopback0 force

PE2	P2	ASBR2
vrf definition vrf1 rd 300:1 route-target export 300:1 route-target import 300:1 route-target import 200:1 ! address-family ipv4 exit-address-family interface Loopback0 ip address 5.1.1.3 255.255.255.255 ip ospf 1 area 0 ! interface Loopback1 vrf forwarding vrf1 ip address 193.1.1.1 255.255.255.255 ip ospf 300 area 0 interface GigabitEthernet1/0/1 no switchport ip address 10.50.1.2 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp ! interface GigabitEthernet1/0/2 no switchport vrf forwarding vrf1 ip address 193.2.1.1 255.255.255.0 ip ospf 300 area 0 router ospf 300 vrf vrf1 router-id 193.1.1.1 nsr nsf redistribute connected redistribute bgp 300 network 193.1.1.1 0.0.0.0 area 0 network 193.2.1.0 0.0.0.255 area 0 ! router ospf 1 router-id 5.1.1.3 nsr nsf redistribute connected router bgp 300 bgp router-id 5.1.1.3 bgp log-neighbor-changes neighbor 5.1.1.1 remote-as 300 neighbor 5.1.1.1 update-source Loopback0 ! address-family ipv4 neighbor 5.1.1.1 activate neighbor 5.1.1.1 send-label exit-address-family ! address-family vpnv4 neighbor 5.1.1.1 activate		

PE2	P2	ASBR2

`neighbor 5.1.1.1`
`send-community extended`
`exit-address-family`
`!`
`address-family ipv4 vrf vrf1`
`redistribute connected`
`redistribute ospf 300`
`maximum-paths ibgp 4`
`exit-address-family`		

Additional References for MPLS VPN InterAS Options

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the MPLS Commands section of the *Command Reference (Catalyst 9500 Series Switches)*

Feature History for MPLS VPN InterAS Options

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	MPLS VPN InterAS Option B	InterAS Options use iBGP and eBGP peering to allow VPNs in different AS to communicate with each other. In an interAS option B network, ASBR ports are connected by one or more interfaces that are enabled to receive MPLS traffic.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 9

Configuring MPLS over GRE

Prerequisites for MPLS over GRE

Ensure that the following routing protocols are configured and working properly.

- Label Distribution Protocol (LDP)—for MPLS label distribution.
- Routing protocol (ISIS or OSFP) between the core devices P1-P-P2
- MPLS between PE1-P1 and PE2-P2
- Since the ingress traffic enters the IP core from MPLS network and egress traffic leaves the IP core to enter the MPLS network, it is recommended to use QoS group value for defining QoS policies as we traverse the protocol boundary.

Restrictions for MPLS over GRE

- GRE Tunneling :
  - L2VPN over mGRE and L3VPN over mGRE is not supported.
  - The tunnel source can only be a loopback or a Layer 3 interface. These interfaces could either be physical interfaces or etherchannels.
  - Tunnel interface supports Static Routes, Enhanced Interior Gateway Routing Protocol (EIGRP) and Open Shortest Path First (OSPF) routing protocols.
  - GRE Options - Sequencing, Checksum and Source Route are not supported.

- IPv6 generic routing encapsulation (GRE) is not supported.
- Carrier Supporting Carrier (CSC) is not supported.
- Tunnel source cannot be a subinterface.

Information About MPLS over GRE

The MPLS over GRE feature provides a mechanism for tunneling Multiprotocol Label Switching (MPLS) packets over a non-MPLS network. This feature allows you to create a generic routing encapsulation (GRE) tunnel across a non-MPLS network. The MPLS packets are encapsulated within the GRE tunnel packets, and the encapsulated packets traverse the non-MPLS network through the GRE tunnel. When GRE tunnel packets are received at the other side of the non-MPLS network, the GRE tunnel packet header is removed and the inner MPLS packet is forwarded to its final destination. The core network between the end-points of the GRE tunnel uses ISIS or OSPF routing protocol whereas the GRE tunnel uses OSPF or EIGRP.

PE-to-PE Tunneling

The provider-edge-to-provider-edge (PE-to-PE) tunneling configuration provides a scalable way to connect multiple customer networks across a non-MPLS network. With this configuration, traffic that is destined to multiple customer networks is multiplexed through a single generic routing encapsulation (GRE) tunnel.

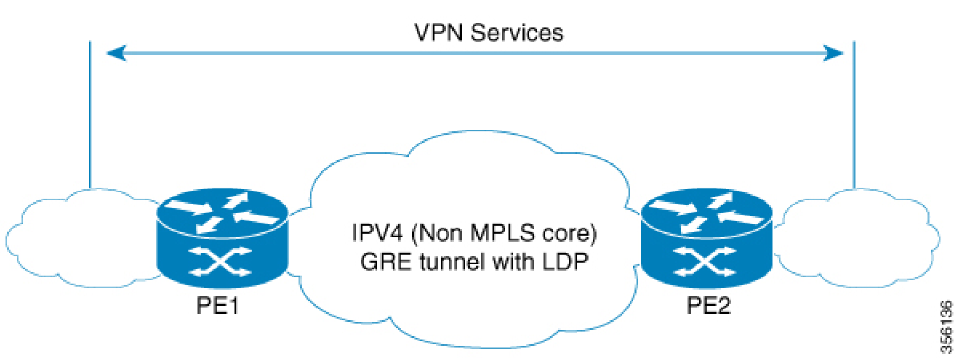
Note A similar nonscalable alternative is to connect each customer network through separate GRE tunnels (for example, connecting one customer network to each GRE tunnel).

The PE device on one side of the non-MPLS network uses the routing protocols (that operate within the non-MPLS network) to learn about the PE device on the other side of the non-MPLS network. The learned routes that are established between the PE devices are then stored in the main or default routing table.

The opposing PE device uses OSPF or EIGRP to learn about the routes that are associated with the customer networks that are behind the PE devices. These learned routes are not known to the non-MPLS network.

The following figure shows an end-to-end IP core from one PE device to another through the GRE tunnel that spans the non-MPLS network.

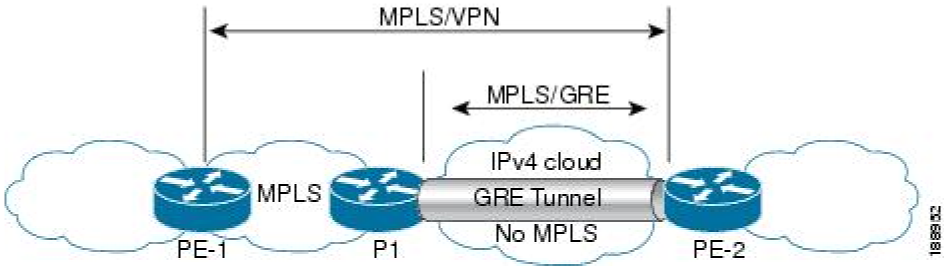
Figure 9: PE-to-PE Tunneling



P-to-PE Tunneling

The provider-to-provider-edge (P-to-PE) tunneling configuration provides a way to connect a PE device (P1) to a Multiprotocol Label Switching (MPLS) segment (PE-2) across a non-MPLS network. In this configuration, MPLS traffic that is destined to the other side of the non-MPLS network is sent through a single generic routing encapsulation (GRE) tunnel.

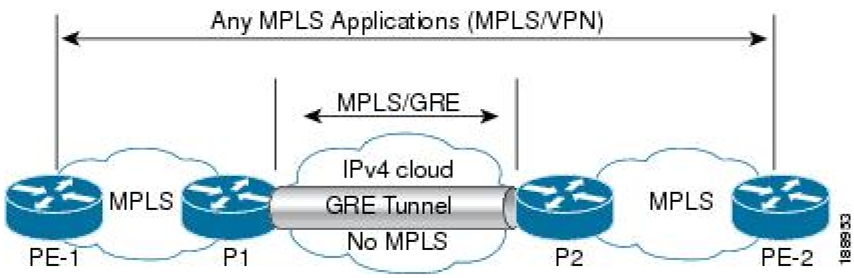
Figure 10: P-to-PE Tunneling



P-to-P Tunneling

As shown in the figure below, the provider-to-provider (P-to-P) configuration provides a method of connecting two Multiprotocol Label Switching (MPLS) segments (P1 to P2) across a non-MPLS network. In this configuration, MPLS traffic that is destined to the other side of the non-MPLS network is sent through a single generic routing encapsulation (GRE) tunnel.

Figure 11: P-to-P Tunneling



How to Configure MPLS over GRE

The following section provides the various configuration steps for MPLS over GRE:

Configuring the MPLS over GRE Tunnel Interface

To configure the MPLS over GRE feature, you must create a generic routing encapsulation (GRE) tunnel to span the non-MPLS networks. You must perform the following procedure on the devices located at both ends of the GRE tunnel.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *tunnel-number* **Example:** `Device(config)# interface tunnel 1`	Creates a tunnel interface and enters interface configuration mode.
Step 4	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 10.0.0.1 255.255.255.0`	Assigns an IP address to the tunnel interface.
Step 5	**tunnel source** *source-address* **Example:** `Device(config-if)# tunnel source 10.1.1.1`	Specifies the tunnel's source IP address.
Step 6	**tunnel destination** *destination-address* **Example:** `Device(config-if)# tunnel destination 10.1.1.2`	Specifies the tunnel's destination IP address.
Step 7	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables Multiprotocol Label Switching (MPLS) on the tunnel's physical interface.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

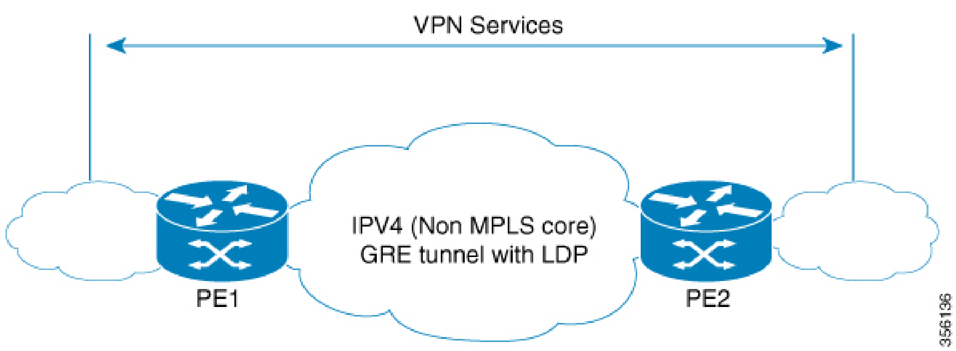
Configuration Examples for MPLS over GRE

The following section provides configuration examples for MPLS over GRE:

Example: PE-to-PE Tunneling

The following shows basic MPLS configuration on two Provider Edge (PE) devices, PE-to-PE tunneling, which use GRE tunnel to send traffic over non-MPLS network.

Figure 12: Topology for PE-to-PE Tunneling



PE1 Configuration

```
!
mpls ip
!
interface loopback 10
ip address 11.2.2.2 255.255.255.255
ip router isis
!
interface GigabitEthernet 1/1/1
ip address 1.1.1.1 255.255.255.0
ip router isis
!
interface Tunnel 1
ip address 10.0.0.1 255.255.255.0
ip ospf 1 are 0
tunnel source 11.2.2.2
tunnel destination 11.1.1.1
mpls ip
!
interface Vlan701
ip address 65.1.1.1 255.255.255.0
ip ospf 1 area 0
!
```

PE2 Configuration

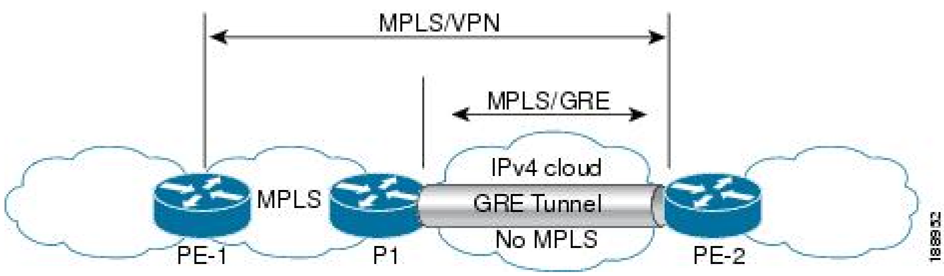
```
!
mpls ip
!
interface loopback 10
```

```
ip address 11.1.1.1 255.255.255.255
ip router isis
!
interface GigabitEthernet 1/1/1
ip address 2.1.1.1 255.255.255.0
ip router isis
!
interface Tunnel 1
ip address 10.0.0.2 255.255.255.0
ip ospf 1 are 0
tunnel source 11.1.1.1
tunnel destination 11.2.2.2
mpls ip
!
interface Vlan701
ip address 75.1.1.1 255.255.255.0
ip ospf 1 area 0
!
```

Example: P-to-PE Tunneling

The following shows basic MPLS configuration on two Provider (P) devices, P-to-PE tunneling, which use GRE tunnel to send traffic over non-MPLS network.

Figure 13: Topology for P-to-PE Tunneling



PE1 Configuration

```
!
mpls ip
!
interface GigabitEthernet 1/1/1
ip address 3.1.1.2 255.255.255.0
ip ospf 1 are 0
mpls ip
!
interface Vlan701
ip address 75.1.1.1 255.255.255.0
ip ospf 1 area 0
!
```

P1 Configuration

```
!
mpls ip
!
interface loopback 10
ip address 11.2.2.2 255.255.255.255
```

```
ip router isis
!
interface GigabitEthernet 1/1/1
ip address 1.1.1.1 255.255.255.0
ip router isis
!
interface GigabitEthernet 1/1/2
ip address 3.1.1.1 255.255.255.0
ip ospf 1 are 0
mpls ip
!
interface Tunnel 1
ip address 10.0.0.1 255.255.255.0
ip ospf 1 are 0
tunnel source 11.2.2.2
tunnel destination 11.1.1.1
mpls ip
!
```

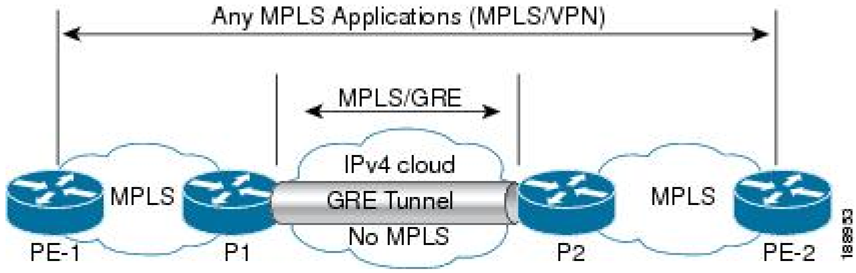
PE2 Configuration

```
!
mpls ip
!
interface loopback 10
ip address 11.1.1.1 255.255.255.255
ip router isis
!
interface GigabitEthernet 1/1/1
ip address 2.2.1.1 255.255.255.0
ip router isis
!
interface Tunnel 1
ip address 10.0.0.2 255.255.255.0
ip ospf 1 are 0
tunnel source 11.1.1.1
tunnel destination 11.2.2.2
mpls ip
!
interface Vlan701
ip address 75.1.1.1 255.255.255.0
ip ospf 1 area 0
!
```

Example: P-to-P Tunneling

The following example shows basic MPLS configuration on two Provider (P) devices, P-to-P tunneling, which use GRE tunnel to send traffic over non-MPLS network.

Figure 14: Topology for P-to-P Tunneling



P1 Configuration

```
!
interface Loopback10
 ip address 10.1.1.1 255.255.255.255
 ip router isis
!
interface Tunnel10
 ip address 10.10.10.1 255.255.255.252
 ip ospf 1 area 0
 mpls ip
 tunnel source 10.1.1.1
 tunnel destination 10.2.1.1
```

P2 Configuration

```
!
interface Tunnel10
 ip address 10.10.10.2 255.255.255.252
 ip ospf 1 area 0
 mpls ip
 tunnel source 10.2.1.1
 tunnel destination 10.1.1.1
!
interface Loopback10
 ip address 10.2.1.1 255.255.255.255
 ip router isis
```

Additional References for MPLS over GRE

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the MPLS Commands section of the *Command Reference (Catalyst 9500 Series Switches)*

Feature History for MPLS over GRE

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	MPLS over GRE	MPLS over GRE feature provides a mechanism for tunneling Multiprotocol Label Switching (MPLS) packets over non-MPLS networks by creating a generic routing encapsulation (GRE) tunnel. The MPLS packets are encapsulated within the GRE tunnel packets, and the encapsulated packets traverse the non-MPLS network through the GRE tunnel. When GRE tunnel packets are received at the other side of the non-MPLS network, the GRE tunnel packet header is removed and the inner MPLS packet is forwarded to its final destination.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/

http://www.cisco.com/go/cfn.

CHAPTER 10

MPLS QoS: Classifying and Marking EXP

Classifying and Marking MPLS EXP

The QoS EXP Matching feature allows you to classify and mark network traffic by modifying the Multiprotocol Label Switching (MPLS) experimental bits (EXP) field. This module contains conceptual information and the configuration tasks for classifying and marking network traffic using the MPLS EXP field.

Prerequisites for Classifying and Marking MPLS EXP

• The switch must be configured as an MPLS provider edge (PE) or provider (P) router, which can include the configuration of a valid label protocol and underlying IP routing protocols.

Restrictions for Classifying and Marking MPLS EXP

• MPLS classification and marking can only occur in an operational MPLS Network.

• If a packet is classified by IP type of service (ToS) or class of service (CoS) at ingress, it cannot be reclassified by MPLS EXP at egress (imposition case). However, if a packet is classified by MPLS at ingress it can be reclassified by IP ToS, CoS, or Quality of Service (QoS) group at egress (disposition case).

• To apply QoS on traffic across protocol boundaries, use QoS-group. You can classify and assign ingress traffic to the QoS-group. Thereafter, you can the QoS-group at egress to classify and apply QoS.

• If a packet is encapsulated in MPLS, the MPLS payload cannot be checked for other protocols such as IP for classification or marking. Only MPLS EXP marking affects packets encapsulated by MPLS.

Information About Classifying and Marking MPLS EXP

This section provides information about classifying and marking MPLS EXP:

Classifying and Marking MPLS EXP Overview

The QoS EXP Matching feature allows you to organize network traffic by setting values for the MPLS EXP field in MPLS packets. By choosing different values for the MPLS EXP field, you can mark packets so that packets have the priority that they require during periods of congestion. Setting the MPLS EXP value allows you to:

- Classify traffic

  The classification process selects the traffic to be marked. Classification accomplishes this by partitioning traffic into multiple priority levels, or classes of service. Traffic classification is the primary component of class-based QoS provisioning. For more information, see the “Classifying Network Traffic” module.

- Police and mark traffic

  Policing causes traffic that exceeds the configured rate to be discarded or marked to a different drop level. Marking traffic is a way to identify packet flows to differentiate them. Packet marking allows you to partition your network into multiple priority levels or classes of service. For more information, see the “Marking Network Traffic” module.

MPLS Experimental Field

The MPLS experimental bits (EXP) field is a 3-bit field in the MPLS header that you can use to define the QoS treatment (per-hop behavior) that a node should give to a packet. In an IP network, the DiffServ Code Point (DSCP) (a 6-bit field) defines a class and drop precedence. The EXP bits can be used to carry some of the information encoded in the IP DSCP and can also be used to encode the dropping precedence.

By default, Cisco IOS Software copies the three most significant bits of the DSCP or the IP precedence of the IP packet to the EXP field in the MPLS header. This action happens when the MPLS header is initially imposed on the IP packet. However, you can also set the EXP field by defining a mapping between the DSCP or IP precedence and the EXP bits. This mapping is configured using the **set mpls experimental** or **police** commands. For more information, see the “How to Classify and Mark MPLS EXP” section.

Note A policy map configured with **set ip dscp** is not supported on the provider edge device because the policy action for MPLS label imposition node should be based on **set mpls experimental imposition** value. However, a policy map with action **set ip dscp** is supported when both the ingress and egress interfaces are Layer 3 ports.

You can perform MPLS EXP marking operations using table-maps. It is recommended to assign QoS-group to a different class of traffic in ingress policy and translate QoS-group to DSCP and EXP markings in egress policy using table-map.

Benefits of MPLS EXP Classification and Marking

If a service provider does not want to modify the value of the IP precedence field in packets transported through the network, they can use the MPLS EXP field value to classify and mark IP packets.

By choosing different values for the MPLS EXP field, you can mark critical packets so that those packets have priority if network congestion occurs.

How to Classify and Mark MPLS EXP

This section provides information about how to classify and mark MPLS EXP:

Classifying MPLS Encapsulated Packets

You can use the **match mpls experimental topmost** command to define traffic classes based on the packet EXP values, inside the MPLS domain. You can use these classes to define services policies to mark the EXP traffic using the **police** command.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**class-map** [**match-all** | **match-any**] *class-map-name* **Example:** `Device(config)# class-map exp3`	Creates a class map to be used for matching traffic to a specified class, and enters class-map configuration mode. • Enter the class map name.
Step 4	**match mpls experimental topmost** *mpls-exp-value* **Example:** `Device(config-cmap)# match mpls experimental topmost 3`	Specifies the match criteria. **Note** The **match mpls experimental topmost** command classifies traffic on the basis of the EXP value in the topmost label header.
Step 5	**end** **Example:** `Device(config-cmap)# end`	(Optional) Returns to privileged EXEC mode.

Marking MPLS EXP on the Outermost Label

Perform this task to set the value of the MPLS EXP field on imposed label entries.

Before you begin

In typical configurations, marking MPLS packets at imposition is used with ingress classification on IP ToS or CoS fields.

Note For IP imposition marking, the IP precedence value is copied to the MPLS EXP value by default.

Note The egress policy on provider edge works with MPLS EXP class match, only if there is a remarking policy at ingress. The provider edge at ingress is an IP interface and only DSCP value is trusted by default. If you do not configure remarking policy at ingress the label for queueing is generated based on DSCP value and not MPLS EXP value. However, a transit provider router works without configuring remarking policy at ingress as the router works on MPLS interfaces.

Note The **set mpls experimental imposition** command works only on packets that have new or additional MPLS labels added to them.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map mark-up-exp-2`	Specifies the name of the policy map to be created and enters policy-map configuration mode. • Enter the policy map name.
Step 4	**class** *class-map-name* **Example:** `Device(config-pmap)# class prec012`	Creates a class map to be used for matching traffic to a specified class, and enters class-map configuration mode. • Enter the class map name.
Step 5	**set mpls experimental imposition** *mpls-exp-value* **Example:** `Device(config-pmap-c)# set mpls experimental imposition 2`	Sets the value of the MPLS EXP field on top label.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-pmap-c)# end`	(Optional) Returns to privileged EXEC mode.

Marking MPLS EXP on Label Switched Packets

Perform this task to set the MPLS EXP field on label switched packets.

Before you begin

Note The **set mpls experimental topmost** command marks EXP for the outermost label of MPLS traffic. Due to this marking at ingress policy, the egress policy must include classification based on the MPLS EXP values.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map mark-up-exp-2`	Specifies the name of the policy map to be created and enters policy-map configuration mode. • Enter the policy map name.
Step 4	**class** *class-map-name* **Example:** `Device(config-pmap)# class-map exp012`	Creates a class map to be used for matching traffic to a specified class, and enters class-map configuration mode. • Enter the class map name.
Step 5	**set mpls experimental topmost** *mpls-exp-value* **Example:** `Device(config-pmap-c)# set mpls experimental topmost 2`	Sets the MPLS EXP field value in the topmost label on the output interface.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-pmap-c)# end`	(Optional) Returns to privileged EXEC mode.

Configuring Conditional Marking

To conditionally set the value of the MPLS EXP field on all imposed label, perform the following task:

Before you begin

Note The **set-mpls-exp-topmost-transmit** action affects MPLS encapsulated packets only. The **set-mpls-exp-imposition-transmit** action affects any new labels that are added to the packet.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Device(config)# policy-map ip2tag`	Specifies the name of the policy map to be created and enters policy-map configuration mode. • Enter the policy map name.
Step 4	**class** *class-map-name* **Example:** `Device(config-pmap)# class iptcp`	Creates a class map to be used for matching traffic to a specified class, and enters policy-map class configuration mode. • Enter the class map name.
Step 5	**police cir** *bps* **bc pir** *bps* **be** **Example:** `Device(config-pmap-c)# police cir 1000000 pir 2000000`	Defines a policer for classified traffic and enters policy-map class police configuration mode.

	Command or Action	Purpose
Step 6	**conform-action transmit** **Example:** `Device(config-pmap-c-police)# conform-action transmit 3`	Defines the action to take on packets that conform to the values specified by the policer. • In this example, if the packet conforms to the committed information rate (cir) or is within the conform burst (bc) size, the MPLS EXP field is set to 3.
Step 7	**exceed-action set-mpls-exp-topmost-transmit dscp table** *dscp-table-value* **Example:** `Device(config-pmap-c-police)# exceed-action set-mpls-exp-topmost-transmit dscp table dscp2exp`	Defines the action to take on packets that exceed the values specified by the policer.
Step 8	**violate-action drop** **Example:** `Device(config-pmap-c-police)# violate-action drop`	Defines the action to take on packets whose rate exceeds the peak information rate (pir) and is outside the bc and be ranges. • You must specify the exceed action before you specify the violate action. • In this example, if the packet rate exceeds the pir rate and is outside the bc and be ranges, the packet is dropped.
Step 9	**end** **Example:** `Device(config-pmap-c-police)# end`	(Optional) Returns to privileged EXEC mode.

Configuration Examples for Classifying and Marking MPLS EXP

This section provides configuration examples for classifying and marking MPLS EXP:

Example: Classifying MPLS Encapsulated Packets

Defining an MPLS EXP Class Map

The following example defines a class map named exp3 that matches packets that contains MPLS experimental value 3:

```
Device(config)# class-map exp3
Device(config-cmap)# match mpls experimental topmost 3
Device(config-cmap)# exit
```

Defining a Policy Map and Applying the Policy Map to an Ingress Interface

The following example uses the class map created in the example above to define a policy map. This example also applies the policy map to a physical interface for ingress traffic.

```
Device(config)# policy-map change-exp-3-to-2
Device(config-pmap)# class exp3
Device(config-pmap-c)# set mpls experimental topmost 2
Device(config-pmap)# exit
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# service-policy input change-exp-3-to-2
Device(config-if)# exit
```

Defining a Policy Map and Applying the Policy Map to an Egress Interface

The following example uses the class map created in the example above to define a policy map. This example also applies the policy map to a physical interface for egress traffic.

```
Device(config)# policy-map WAN-out
Device(config-pmap)# class exp3
Device(config-pmap-c)# shape average 10000000
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# service-policy output WAN-out
Device(config-if)# exit
```

Example: Marking MPLS EXP on Outermost Label

Defining an MPLS EXP Imposition Policy Map

The following example defines a policy map that sets the MPLS EXP imposition value to 2 based on the IP precedence value of the forwarded packet:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# class-map prec012
Device(config-cmap)# match ip prec 0 1 2
Device(config-cmap)# exit
Device(config)# policy-map mark-up-exp-2
Device(config-pmap)# class prec012
Device(config-pmap-c)# set mpls experimental imposition 2
Device(config-pmap-c)# exit
Device(config-pmap)# exit
```

Applying the MPLS EXP Imposition Policy Map to a Main Interface

The following example applies a policy map to Gigabit Ethernet interface 0/0/0:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# service-policy input mark-up-exp-2
Device(config-if)# exit
```

Example: Marking MPLS EXP on Label Switched Packets

Defining an MPLS EXP Label Switched Packets Policy Map

The following example defines a policy map that sets the MPLS EXP topmost value to 2 according to the MPLS EXP value of the forwarded packet:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# class-map exp012
Device(config-cmap)# match mpls experimental topmost 0 1 2
Device(config-cmap)# exit
Device(config-cmap)# policy-map mark-up-exp-2
Device(config-pmap)# class exp012
Device(config-pmap-c)# set mpls experimental topmost 2
Device(config-pmap-c)# exit
Device(config-pmap)# exit
```

Applying the MPLS EXP Label Switched Packets Policy Map to a Main Interface

The following example shows how to apply the policy map to a main interface:

```
Switch# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# service-policy input mark-up-exp-2
Device(config-if)# exit
```

Example: Configuring Conditional Marking

The example in this section creates a policer for the **iptcp** class, which is part of the **ip2tag** policy map, and attaches the policy map to the Gigabit Ethernet interface.

```
Device(config)# policy-map ip2tag
Device(config-pmap)# class iptcp
Device(config-pmap-c)# police cir 1000000 pir 2000000
Device(config-pmap-c-police)# conform-action transmit
Device(config-pmap-c-police)# exceed-action set-mpls-exp-imposition-transmit 2
Device(config-pmap-c-police)# violate-action drop
Device(config-pmap-c-police)# exit
Device(config-pmap-c)# exit
Device(config-pmap)# exit
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# service-policy input ip2tag
```

Additional References

Related Documents

Related Topic	Document Title
QoS commands	*Cisco IOS Quality of Service Solutions Command Reference*

Feature History for QoS MPLS EXP

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	QoS MPLS EXP	The QoS EXP Matching feature allows you to classify, mark and queue network traffic by modifying the Multiprotocol Label Switching (MPLS) experimental bits (EXP) field.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 11

Configuring MPLS Static Labels

MPLS Static Labels

This document describes the Cisco MPLS Static Labels feature. The MPLS Static Labels feature provides the means to configure statically:

- The binding between a label and an IPv4 prefix
- The contents of an LFIB crossconnect entry

Prerequisites for MPLS Static Labels

The network must support the following Cisco IOS features before you enable MPLS static labels:

- Multiprotocol Label Switching (MPLS)
- Cisco Express Forwarding

Restrictions for MPLS Static Labels

- The trouble shooting process for MPLS static labels is complex.
- On a provider edge (PE) router for MPLS VPNs, there’s no mechanism for statically binding a label to a customer network prefix (VPN IPv4 prefix).
- MPLS static crossconnect labels remain in the LFIB even if the router to which the entry points goes down.
- MPLS static crossconnect mappings remain in effect even with topology changes.
- MPLS static labels aren’t supported for label-controlled Asynchronous Transfer Mode (lc-atm).
- MPLS static bindings aren’t supported for local prefixes.

Information About MPLS Static Labels

MPLS Static Labels Overview

Generally, label switching routers (LSRs) dynamically learn the labels they should use to label-switch packets. They do this by means of label distribution protocols that include:

- Label Distribution Protocol (LDP), the Internet Engineering Task Force (IETF) standard, used to bind labels to network addresses.
- Resource Reservation Protocol (RSVP) used to distribute labels for traffic engineering (TE)
- Border Gateway Protocol (BGP) used to distribute labels for Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs)

To use a learned label to label-switch packets, an LSR installs the label into its Label Forwarding Information Base (LFIB).

The MPLS Static Labels feature provides the means to configure statically:

- The binding between a label and an IPv4 prefix
- The contents of an LFIB crossconnect entry

Benefits of MPLS Static Labels

Static Bindings Between Labels and IPv4 Prefixes

You can configure static bindings between labels and IPv4 prefixes to support MPLS hop-by-hop forwarding through neighbor routers that don’t implement LDP label distribution.

Static Crossconnects

You can configure static crossconnects to support MPLS Label Switched Path (LSP) midpoints when neighbor routers don’t implement either the LDP or RSVP label distribution, but do implement an MPLS forwarding path.

How to Configure MPLS Static Labels

Configuring MPLS Static Prefix Label Bindings

To configure MPLS static prefix/label bindings, use the following commands beginning in global configuration mode:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mpls label range** *min-label max-label* [**static** *min-static-label max-static-label*] **Example:** `Device(config)# mpls label range 200 100000 static 16 199`	Specifies a range of labels for use with MPLS Static Labels feature. (Default is no labels reserved for static assignment.)
Step 4	**mpls static binding ipv4** *prefix mask* [**input**| **output** *nexthop*] label **Example:** `Device(config)# mpls static binding ipv4 10.0.0.0 255.0.0.0 55`	Specifies static binding of labels to IPv4 prefixes. Bindings specified are installed automatically in the MPLS forwarding table as routing demands.

Verifying MPLS Static Prefix Label Bindings

To verify the configuration for MPLS static prefix/label bindings, use this procedure:

Procedure

Step 1 Enter **show mpls label range** command. The output shows that the new label ranges do not take effect until a reload occurs:

Example:

```
Device# show mpls label range

Downstream label pool: Min/Max label: 16/100000
   [Configured range for next reload: Min/Max label: 200/100000]
Range for static labels: Min/Max/Number: 16/199
```

The following output from the **show mpls label range** command, executed after a reload, indicates that the new label ranges are in effect:

Example:

```
Device# show mpls label range

Downstream label pool: Min/Max label: 200/100000
Range for static labels: Min/Max/Number: 16/199
```

Step 2 Enter the **show mpls static binding ipv4** command to show the configured static prefix/label bindings:

Example:

```
Device# show mpls static binding ipv4
```

```
10.17.17.17/32: Incoming label: 251 (in LIB)
  Outgoing labels:
     10.0.0.1                  18
10.18.18.18/32: Incoming label: 201 (in LIB)
  Outgoing labels:
10.0.0.1 implicit-null
```

Step 3 Use the **show mpls forwarding-table** command to determine which static prefix/label bindings are currently in use for MPLS forwarding.

Example:

```
Device# show mpls forwarding-table
Local  Outgoing    Prefix           Bytes tag  Outgoing    Next Hop
tag    tag or VC   or Tunnel Id     switched   interface
201    Pop tag     10.18.18.18/32   0          PO1/1/0     point2point
       2/35        10.18.18.18/32   0          AT4/1/0.1   point2point
251    18          10.17.17.17/32   0          PO1/1/0     point2point
```

Monitoring and Maintaining MPLS Static Labels

To monitor and maintain MPLS static labels, use one or more of the following commands:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Devie> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**show mpls forwarding-table** **Example:** `Device# show mpls forwarding-table`	Displays the contents of the MPLS LFIB.
Step 3	**show mpls label range** **Example:** `Device# show mpls label range`	Displays information about the static label range.
Step 4	**show mpls static binding ipv4** **Example:** `Device# show mpls static binding ipv4`	Displays information about the configured static prefix/label bindings.
Step 5	**show mpls static crossconnect** **Example:** `Device# show mpls static crossconnect`	Displays information about the configured crossconnects.

Configuration Examples for MPLS Static Labels

Example Configuring MPLS Static Prefixes Labels

In the following output, the **mpls label range** command reconfigures the range used for dynamically assigned labels 16–100000 to 200–100000. It configures a static label range of 16–199.

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# mpls label range 200 100000 static 16 199
% Label range changes take effect at the next reload.
Router(config)# end
```

In the following output, the **show mpls label range** command indicates that the new label ranges don’t take effect until a reload occurs:

```
Device# show mpls label range

Downstream label pool: Min/Max label: 16/100000
   [Configured range for next reload: Min/Max label: 200/100000]
Range for static labels: Min/Max/Number: 16/199
```

In the following output, the **show mpls label range** command, executed after a reload, indicates that the new label ranges are in effect:

```
Device# show mpls label range

Downstream label pool: Min/Max label: 200/100000
Range for static labels: Min/Max/Number: 16/199
```

In the following output, the **mpls static binding ipv4** commands configure static prefix/label bindings. They also configure input (local) and output (remote) labels for various prefixes:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# mpls static binding ipv4 10.0.0.0 255.0.0.0 55
Device(config)# mpls static binding ipv4 10.0.0.0 255.0.0.0 output 10.0.0.66 2607
Device(config)# mpls static binding ipv4 10.6.0.0 255.255.0.0 input 17
Device(config)# mpls static binding ipv4 10.0.0.0 255.0.0.0 output 10.13.0.8 explicit-null
Device(config)# end
```

In the following output, the **show mpls static binding ipv4** command displays the configured static prefix/label bindings:

```
Device# show mpls static binding ipv4

10.0.0.0/8: Incoming label: none;
  Outgoing labels:
10.13.0.8           explicit-null
10.0.0.0/8: Incoming label: 55 (in LIB)
  Outgoing labels:
     10.0.0.66              2607
10.66.0.0/16: Incoming label: 17 (in LIB)
  Outgoing labels:  None
```

Additional References

Related Documents

Related Topic	Document Title
MPLS commands	*Multiprotocol Label Switching Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature. Support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature History for MPLS Static Labels

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	MPLS Static Labels	The MPLS Static Labels feature provides the means to configure the binding between a label and an IPv4 prefix statically. The following commands were introduced or modified: **debug mpls static binding**, **mpls label range**, **mpls static binding ipv4**, **show mpls label range**, **show mpls static binding ipv4**
Cisco IOS XE Gibraltar 16.11.1	MPLS Static Labels	Support for this feature was introduced only on the C9500-32C, C9500-32QC, C9500-48Y4C, and C9500-24Y4C models of the Cisco Catalyst 9500 Series Switches.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 12

Configuring Virtual Private LAN Service (VPLS) and VPLS BGP-Based Autodiscovery

Configuring VPLS

The following sections provide information about how to configure VPLS.

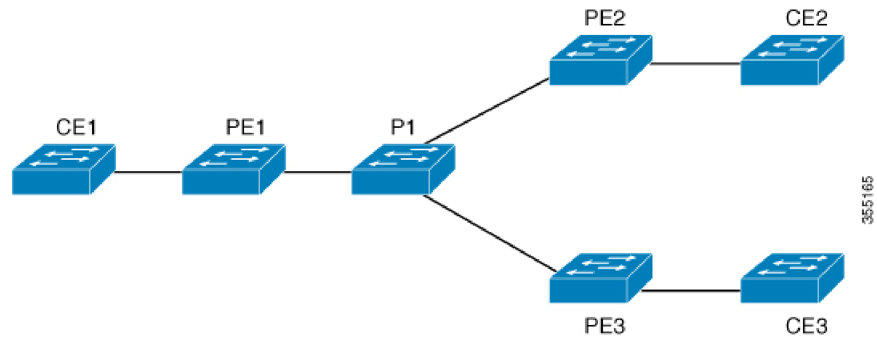
Information About VPLS

VPLS Overview

VPLS (Virtual Private LAN Service) enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider. From the enterprise perspective, the service provider's public network looks like one giant Ethernet LAN. For the service provider, VPLS provides an opportunity to deploy another revenue-generating service on top of their existing network without major capital expenditures. Operators can extend the operational life of equipment in their network.

Virtual Private LAN Service (VPLS) uses the provider core to join multiple attachment circuits together to simulate a virtual bridge that connects the multiple attachment circuits together. From a customer point of view, there is no topology for VPLS. All of the CE devices appear to connect to a logical bridge emulated by the provider core.

Figure 15: VPLS Topology



Full-Mesh Configuration

The full-mesh configuration requires a full mesh of tunnel label switched paths (LSPs) between all the PEs that participate in the VPLS. With full-mesh, signaling overhead and packet replication requirements for each provisioned VC on a PE can be high.

You set up a VPLS by first creating a virtual forwarding instance (VFI) on each participating PE router. The VFI specifies the VPN ID of a VPLS domain, the addresses of other PE devices in the domain, and the type of tunnel signaling and encapsulation mechanism for each peer PE router.

The set of VFIs formed by the interconnection of the emulated VCs is called a VPLS instance; it is the VPLS instance that forms the logic bridge over a packet switched network. The VPLS instance is assigned a unique VPN ID.

The PE devices use the VFI to establish a full-mesh LSP of emulated VCs to all the other PE devices in the VPLS instance. PE devices obtain the membership of a VPLS instance through static configuration using the Cisco IOS CLI.

The full-mesh configuration allows the PE router to maintain a single broadcast domain. Thus, when the PE router receives a broadcast, multicast, or unknown unicast packet on an attachment circuit, it sends the packet out on all other attachment circuits and emulated circuits to all other CE devices participating in that VPLS instance. The CE devices see the VPLS instance as an emulated LAN.

To avoid the problem of a packet looping in the provider core, the PE devices enforce a "split-horizon" principle for the emulated VCs. That means if a packet is received on an emulated VC, it is not forwarded on any other emulated VC.

After the VFI has been defined, it needs to be bound to an attachment circuit to the CE device.

The packet forwarding decision is made by looking up the Layer 2 virtual forwarding instance (VFI) of a particular VPLS domain.

A VPLS instance on a particular PE router receives Ethernet frames that enter on specific physical or logical ports and populates a MAC table similarly to how an Ethernet switch works. The PE router can use the MAC address to switch those frames into the appropriate LSP for delivery to the another PE router at a remote site.

If the MAC address is not in the MAC address table, the PE router replicates the Ethernet frame and floods it to all logical ports associated with that VPLS instance, except the ingress port where it just entered. The PE router updates the MAC table as it receives packets on specific ports and removes addresses not used for specific periods.

Restrictions for VPLS

- Layer 2 protocol tunneling configuration is not supported
- Virtual Circuit Connectivity Verification (VCCV) ping with explicit null is not supported.
- The switch is supported if configured only as a spoke in hierarchical Virtual Private LAN Services (VPLS) and not as a hub.
- Layer 2 VPN interworking functions are not supported.
- **ip unnumbered** command is not supported in Multiprotocol Label Switching (MPLS) configuration.
- Virtual Circuit (VC) statistics are not displayed for flood traffic in the output of **show mpls l2 vc vcid detail** command.
- Dot1q tunnel configuration is not supported in the attachment circuit.

Configuring Layer 2 PE Device Interfaces to CE Devices

You must configure Layer 2 PE device interfaces to CE devices. You can either configure 802.1Q trunks on the PE device for tagged traffic from a CE device or configure 802.1Q access ports on the PE device for untagged traffic from a CE device. The following sections provides configuration information for both.

Configuring 802.1Q Trunks on a PE Device for Tagged Traffic from a CE Device

To configure 802.1Q trunks on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface TenGigabitEthernet1/0/24`	Defines the interface to be configured as a trunk, and enters interface configuration mode.
Step 4	**no ip address** *ip_address mask* [**secondary**] **Example:** `Device(config-if)# no ip address`	Disables IP processing and enters interface configuration mode.

	Command or Action	Purpose
Step 5	**switchport** **Example:** `Device(config-if)# switchport`	Modifies the switching characteristics of the Layer 2 switched interface.
Step 6	**switchport trunk encapsulation dot1q** **Example:** `Device(config-if)# switchport trunk encapsulation dot1q`	Sets the switch port encapsulation format to 802.1Q.
Step 7	**switchport trunk allow vlan** *vlan_ID* **Example:** `Device(config-if)# switchport trunk allow vlan 2129`	Sets the list of allowed VLANs.
Step 8	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface to a trunking VLAN Layer 2 interface.
Step 9	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring 802.1Q Access Ports on a PE Device for Untagged Traffic from a CE Device

To configure 802.1Q access ports on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:**	Defines the interface to be configured as a trunk, and enters interface configuration mode.

	Command or Action	Purpose
	Device(config)# **interface TenGigabitEthernet1/0/24**	
Step 4	**no ip address** *ip_address mask* [**secondary**] **Example:** Device(config-if)# **no ip address**	Disables IP processing.
Step 5	**switchport** **Example:** Device(config-if)# **switchport**	Modifies the switching characteristics of the Layer 2 switched interface.
Step 6	**switchport mode access** **Example:** Device(config-if)# **switchport mode access**	Sets the interface type to nontrunking and nontagged single VLAN Layer 2 interface.
Step 7	**switchport access vlan** *vlan_ID* **Example:** Device(config-if)# **switchport access vlan 2129**	Sets the VLAN when the interface is in access mode.
Step 8	**end** **Example:** Device(config-if)# **end**	Returns to privileged EXEC mode.

Configuring Layer 2 VLAN Instances on a PE Device

Configuring the Layer 2 VLAN interface on the PE device, enables the Layer 2 VLAN instance on the PE device to the VLAN database, to set up the mapping between the VPLS and VLANs.

To configure Layer 2 VLAN instance on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** Device> **enable**	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vlan** *vlan-id* **Example:** `Device(config)# vlan 2129`	Configures a specific VLAN.
Step 4	**interface vlan** *vlan-id* **Example:** `Device(config-vlan)# interface vlan 2129`	Configures an interface on the VLAN.
Step 5	**end** **Example:** `Device(config-vlan)# end`	Returns to privileged EXEC mode.

Configuring MPLS on a PE Device

To configure MPLS on a PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**mpls ip** **Example:** `Device(config)# mpls ip`	Configures MPLS hop-by-hop forwarding.
Step 4	**mpls label protocol ldp** **Example:**	Specifies the default Label Distribution Protocol (LDP) for a platform.

	Command or Action	Purpose
	`Device(config)# mpls label protocol ldp`	
Step 5	**mpls ldp logging neighbor-changes** **Example:** `Device(config)# mpls ldp logging neighbor-changes`	(Optional) Determines logging neighbor changes.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuring VFI on a PE Device

The VFI specifies the VPN ID of a VPLS domain, the addresses of other PE devices in this domain, and the type of tunnel signaling and encapsulation mechanism for each peer device.

To configure VFI and associated VCs on the PE device, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **manual** **Example:** `Device(config)# l2 vfi 2129 manual`	Enables the Layer 2 VFI manual configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 2129`	Configures a VPN ID for a VPLS domain. The emulated VCs bound to this Layer 2 virtual routing and forwarding (VRF) use this VPN ID for signaling. **Note** *vpn-id* is the same as *vlan-id*.

	Command or Action	Purpose
Step 5	**neighbor** *router-id* {**encapsulation mpls**} **Example:** `Device(config-vfi)# neighbor remote-router-id encapsulation mpls`	Specifies the remote peering router ID and the tunnel encapsulation type or the pseudowire (PW) property to be used to set up the emulated VC.
Step 6	**end** **Example:** `Device(config-vfi)# end`	Returns to privileged EXEC mode.

Associating the Attachment Circuit with the VFI on the PE Device

After defining the VFI, you must associate it to one or more attachment circuits.

To associate the attachment circuit with the VFI, perform this procedure:

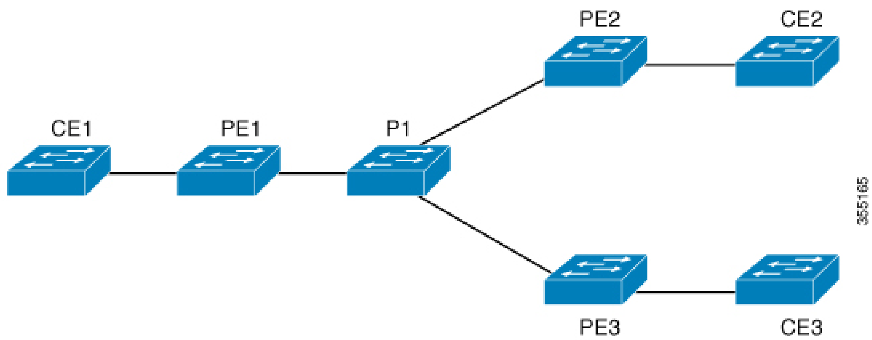
Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface vlan** *vlan-id* **Example:** `Device(config)# interface vlan 2129`	Creates or accesses a dynamic switched virtual interface (SVI). **Note** *vlan-id* is the same as *vpn-id*.
Step 4	**no ip address** **Example:** `Device(config-if)# no ip address`	Disables IP processing. (You can configure a Layer 3 interface for the VLAN if you need to configure an IP address.)
Step 5	**xconnect vfi** *vfi-name* **Example:** `Device(config-if)# xconnect vfi 2129`	Specifies the Layer 2 VFI that you are binding to the VLAN port.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuration Examples for VPLS

Figure 16: VPLS Topology



PE1 Configuration

```
pseudowire-class vpls2129
 encapsulation mpls
!
l2 vfi 2129 manual
 vpn id 2129
 neighbor 44.254.44.44 pw-class vpls2129

 neighbor 188.98.89.98 pw-class vpls2129
!
interface TenGigabitEthernet1/0/24
 switchport trunk allowed vlan 2129
 switchport mode trunk
!
interface Vlan2129
 no ip address
 xconnect vfi 2129
!
```

PE2 Configuration

```
pseudowire-class vpls2129
 encapsulation mpls
 no control-word
!
l2 vfi 2129 manual
 vpn id 2129
 neighbor 1.1.1.72 pw-class vpls2129
 neighbor 188.98.89.98 pw-class vpls2129
!
interface TenGigabitEthernet1/0/47
 switchport trunk allowed vlan 2129
 switchport mode trunk
end
!
interface Vlan2129
 no ip address
 xconnect vfi 2129
!
```

The **show mpls 12transport vc detail** command provides information the virtual circuits.

```
Local interface: VFI 2129 vfi up
  Interworking type is Ethernet
  Destination address: 44.254.44.44, VC ID: 2129, VC status: up
    Output interface: Gi1/0/9, imposed label stack {18 17}
    Preferred path: not configured
    Default path: active
```

```
      Next hop: 177.77.177.2
    Create time: 19:09:33, last status change time: 09:24:14
      Last label FSM state change time: 09:24:14
    Signaling protocol: LDP, peer 44.254.44.44:0 up
      Targeted Hello: 1.1.1.72(LDP Id) -> 44.254.44.44, LDP is UP
      Graceful restart: configured and enabled
      Non stop routing: not configured and not enabled
      Status TLV support (local/remote)   : enabled/supported
        LDP route watch                   : enabled
        Label/status state machine        : established, LruRru
        Last local dataplane   status rcvd: No fault
  Last BFD dataplane     status rcvd: Not sent
        Last BFD peer monitor  status rcvd: No fault
        Last local AC  circuit status rcvd: No fault
        Last local AC  circuit status sent: No fault
        Last local PW i/f circ status rcvd: No fault
        Last local LDP TLV     status sent: No fault
        Last remote LDP TLV    status rcvd: No fault
        Last remote LDP ADJ    status rcvd: No fault
  MPLS VC labels: local 512, remote 17
      Group ID: local n/a, remote 0
      MTU: local 1500, remote 1500
      Remote interface description:
    Sequencing: receive disabled, send disabled
    Control Word: Off
    SSO Descriptor: 44.254.44.44/2129, local label: 512
    Dataplane:
      SSM segment/switch IDs: 20498/20492 (used), PWID: 2
    VC statistics:
      transit packet totals: receive 0, send 0
      transit byte totals:   receive 0, send 0
      transit packet drops:  receive 0, seq error 0, send 0
```

The **show l2vpn atom vc** shows that ATM over MPLS is configured on a VC.

```
pseudowire100005 is up, VC status is up PW type: Ethernet
  Create time: 19:25:56, last status change time: 09:40:37
    Last label FSM state change time: 09:40:37
  Destination address: 44.254.44.44 VC ID: 2129
    Output interface: Gi1/0/9, imposed label stack {18 17}
    Preferred path: not configured
    Default path: active
    Next hop: 177.77.177.2
  Member of vfi service 2129
    Bridge-Domain id: 2129
    Service id: 0x32000003
  Signaling protocol: LDP, peer 44.254.44.44:0 up
    Targeted Hello: 1.1.1.72(LDP Id) -> 44.254.44.44, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: not configured and not enabled
     PWid FEC (128), VC ID: 2129
```

```
      Status TLV support (local/remote)         : enabled/supported
        LDP route watch                         : enabled
        Label/status state machine              : established, LruRru
        Local dataplane status received         : No fault
        BFD dataplane status received           : Not sent
        BFD peer monitor status received        : No fault
        Status received from access circuit     : No fault
        Status sent to access circuit           : No fault
        Status received from pseudowire i/f     : No fault
Status sent to network peer               : No fault
        Status received from network peer       : No fault
        Adjacency status of remote peer         : No fault
  Sequencing: receive disabled, send disabled
  Bindings
    Parameter    Local                          Remote
    ------------ ------------------------------
------------------------------
    Label        512                            17
    Group ID     n/a                            0
    Interface

    MTU          1500                           1500
    Control word off                            off
    PW type      Ethernet                       Ethernet
    VCCV CV type 0x02                           0x02
                   LSPV [2]                       LSPV [2]

    VCCV CC type 0x06                           0x06
                   RA [2], TTL [3]               RA [2], TTL [3]
    Status TLV   enabled                        supported
  SSO Descriptor: 44.254.44.44/2129, local label: 512
  Dataplane:
    SSM segment/switch IDs: 20498/20492 (used), PWID: 2
  Rx Counters
    0 input transit packets, 0 bytes
    0 drops, 0 seq err
  Tx Counters
    0 output transit packets, 0 bytes
    0 drops
```

Configuring VPLS BGP-based Autodiscovery

The following sections provide information about how to configure VPLS BGP-based Autodiscovery.

Information About VPLS BGP-Based Autodiscovery

VPLS BGP Based Autodiscovery

VPLS Autodiscovery enables each Virtual Private LAN Service (VPLS) provider edge (PE) device to discover other PE devices that are part of the same VPLS domain. VPLS Autodiscovery also tracks PE devices when they are added to or removed from a VPLS domain. As a result, with VPLS Autodiscovery enabled, you no longer need to manually configure a VPLS domain and maintain the configuration when a PE device is added or deleted. VPLS Autodiscovery uses the Border Gateway Protocol (BGP) to discover VPLS members and set up and tear down pseudowires in a VPLS domain

BGP uses the Layer 2 VPN (L2VPN) Routing Information Base (RIB) to store endpoint provisioning information, which is updated each time any Layer 2 virtual forwarding instance (VFI) is configured. The prefix and path information is stored in the L2VPN database, which allows BGP to make decisions about the best path. When BGP distributes the endpoint provisioning information in an update message to all its BGP neighbors, this endpoint information is used to configure a pseudowire mesh to support L2VPN-based services.

The BGP autodiscovery mechanism facilitates the configuration of L2VPN services, which are an integral part of the VPLS feature. VPLS enables flexibility in deploying services by connecting geographically dispersed sites as a large LAN over high-speed Ethernet in a robust and scalable IP Multiprotocol Label Switching (MPLS) network.

Enabling VPLS BGP-based Autodiscovery

To enabling VPLS BGP-based autodiscovery, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**l2 vfi** *vfi-name* **autodiscovery** **Example:** `Device(config)# l2 vfi 2128 autodiscovery`	Enables VPLS autodiscovery on a PE device and enters L2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:**	Configures a VPN ID for the VPLS domain.

	Command or Action	Purpose
	`Device(config-vfi)# vpn id 2128`	
Step 5	**end** **Example:** `Device(config-vfi)# end`	Returns to privileged EXEC mode.

Configuring BGP to Enable VPLS Autodiscovery

To configure BGP to enable VPLS autodiscovery, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1000`	Enters router configuration mode for the specified routing process.
Step 4	**no bgp default ipv4-unicast** **Example:**	Disables the IPv4 unicast address family for the BGP routing process.

	Command or Action	Purpose
	`Device(config-router)# no bgp default ipv4-unicast`	**Note** Routing information for the IPv4 unicast address family is advertised by default for each BGP routing session configured using the **neighbor remote-as router** command unless you configure the **no bgp default ipv4-unicast** command before configuring the **neighbor remote-as** command. Existing neighbor configurations are not affected.
Step 5	**bgp log-neighbor-changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 6	**neighbor remote-as** { *ip-address* | *peer-group-name* } **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 44.254.44.44 remote-as 1000`	Adds the IP address or peer group name of the neighbor in the specified autonomous system to the IPv4 multiprotocol BGP neighbor table of the local device. • If the *autonomous-system-number* argument matches the autonomous system number specified in the **router bgp** command, the neighbor is an internal neighbor. • If the *autonomous-system-number* argument does not match the autonomous system number specified in the **router bgp** command, the neighbor is an external neighbor.
Step 7	**neighbor** { *ip-address* | *peer-group-name* } **update-source** *interface-type interface-number* **Example:** `Device(config-router)#  neighbor 44.254.44.44 update-source Loopback300`	(Optional) Configures a device to select a specific source or interface to receive routing table updates.
Step 8	Repeat Steps 6 and 7 to configure other BGP neighbors.	Exits interface configuration mode.
Step 9	**address-family l2vpn** [**vpls**] **Example:**	Specifies the Layer 2 VPN address family and enters address family configuration mode.

	Command or Action	Purpose
	`Device(config-router)# **address-family l2vpn vpls**`	The optional **vpls** keyword specifies that the VPLS endpoint provisioning information is to be distributed to BGP peers.
Step 10	**neighbor** { *ip-address* | *peer-group-name* } **activate** **Example:** `Device(config-router-af)# **neighbor 44.254.44.44 activate**`	Enables the exchange of information with a BGP neighbor.
Step 11	**neighbor** { *ip-address* | *peer-group-name* } **send-community** { **both** | **standard** | **extended** } **Example:** `Device(config-router-af)# **neighbor 44.254.44.44 send-community both**`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 12	Repeat Steps 10 and 11 to activate other BGP neighbors under an L2VPN address family.	
Step 13	**exit-address-family** **Example:** `Device(config-router-af)# **exit-address-family**`	Exits address family configuration mode and returns to router configuration mode.
Step 14	**end** **Example:** `Device(config-router)# **end**`	Exits router configuration mode and returns to privileged EXEC mode.

Configuration Examples for VPLS BGP-AD

PE Configuration

```
router bgp 1000
 bgp log-neighbor-changes
 bgp graceful-restart
 neighbor 44.254.44.44 remote-as 1000
 neighbor 44.254.44.44 update-source Loopback300
!
 address-family l2vpn vpls
  neighbor 44.254.44.44 activate
  neighbor 44.254.44.44 send-community both
 exit-address-family
!
l2 vfi 2128 autodiscovery
 vpn id 2128
interface Vlan2128
 no ip address
 xconnect vfi 2128
!
```

The following is a sample output of **show platform software fed sw 1 matm macTable vlan 2000** command :

```
VLAN  MAC               Type      Seq#   macHandle          siHandle
      diHandle            *a_time *e_time   ports
2000  2852.6134.05c8   0X8002    0      0xffbba312c8       0xffbb9ef938
      0x5154              0         0        Vlan2000
2000  0000.0078.9012   0X1       32627  0xffbb665ec8       0xffbb60b198
      0xffbb653f98        300      278448   Port-channel11
2000  2852.6134.0000   0X1       32651  0xffba15e1a8       0xff454c2328
      0xffbb653f98        300      63       Port-channel11
2000  0000.0012.3456   0X2000001 32655  0xffba15c508       0xff44f9ec98
      0x0                 300      1        2000:33.33.33.33
Total Mac number of addresses:: 4
*a_time=aging_time(secs)  *e_time=total_elapsed_time(secs)
Type:
MAT_DYNAMIC_ADDR     0x1       MAT_STATIC_ADDR       0x2
MAT_CPU_ADDR         0x4       MAT_DISCARD_ADDR      0x8
MAT_ALL_VLANS        0x10      MAT_NO_FORWARD        0x20
MAT_IPMULT_ADDR      0x40      MAT_RESYNC            0x80
MAT_DO_NOT_AGE       0x100     MAT_SECURE_ADDR       0x200
MAT_NO_PORT          0x400     MAT_DROP_ADDR         0x800
MAT_DUP_ADDR         0x1000    MAT_NULL_DESTINATION  0x2000
MAT_DOT1X_ADDR       0x4000    MAT_ROUTER_ADDR       0x8000
MAT_WIRELESS_ADDR    0x10000   MAT_SECURE_CFG_ADDR   0x20000
MAT_OPQ_DATA_PRESENT 0x40000   MAT_WIRED_TUNNEL_ADDR 0x80000
MAT_DLR_ADDR         0x100000  MAT_MRP_ADDR          0x200000
MAT_MSRP_ADDR        0x400000  MAT_LISP_LOCAL_ADDR   0x800000
MAT_LISP_REMOTE_ADDR 0x1000000 MAT_VPLS_ADDR         0x2000000
```

The following is a sample output of **show bgp l2vpn vpls all** command :

```
BGP table version is 6, local router ID is 222.5.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
  r RIB-failure, S Stale, m multipath, b backup-path, f RT-Filter,
  x best-external, a additional-path, c RIB-compressed,
  t secondary path,
 Origin codes: i - IGP, e - EGP, ? – incomplete
RPKI validation codes: V valid, I invalid, N Not found
Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1000:2128
*>   1000:2128:1.1.1.72/96
                0.0.0.0                            32768 ?
*>i  1000:2128:44.254.44.44/96
                44.254.44.44             0    100      0 ?
```

Feature Information for VPLS and VPLS BGP-Based Autodiscovery

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 3: Feature Information for VPLS and VPLS BGP-based Autodiscovery

Feature Name	Releases	Feature Information
Configuring VPLS and VPLS BGP-based Autodiscovery	Cisco IOS XE Everest 16.5.1a	VPLS enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider. VPLS Autodiscovery enables each PE device to discover other PE devices that are part of the same VPLS domain.
Configuring VPLS and VPLS BGP-based Autodiscovery	Cisco IOS XE Fuji 16.9.1	This feature was implemented on Cisco Catalyst 9500 Series Switches - High Performance.

CHAPTER 13

Configuring VPLS MAC Address Withdrawal

Restrictions for VPLS MAC Address Withdrawal

This feature is not supported on the C9500-12Q, C9500-16X, C9500-24Q, C9500-40X models of the Cisco Catalyst 9500 Series Switches.

VPLS MAC Address Withdrawal

The VPLS MAC Address Withdrawal feature provides faster convergence by removing (or unlearning) MAC addresses that have been dynamically learned. A Label Distribution Protocol (LDP)-based MAC address withdrawal message is used for this purpose. A MAC list Type Length Value (TLV) is part of the MAC address withdrawal message.

The **debug mpls ldp messages** and **debug mpls ldp session io** commands support monitoring of MAC address withdrawal messages being exchanged between LDP peers. Any Transport over Multiprotocol Label Switching (AToM) might provide other means to display or monitor MAC address withdrawal messages. The Tag Distribution Protocol (TDP) is not supported because AToM uses only LDP for the MAC address withdrawal message.

PE devices learn the remote MAC addresses and directly attached MAC addresses on customer-facing ports by deriving the topology and forwarding information from packets originating at customer sites. To display the number of MAC address withdrawal messages, enter the **show mpls l2transport vc detail** command, as shown in the following example:

```
Device# show mpls l2transport vc detail

Local interface: VFI TEST VFI up
  MPLS VC type is VFI, interworking type is Ethernet
  Destination address: 10.1.1.1, VC ID: 1000, VC status: up
    Output interface: Se2/0, imposed label stack {17}
    Preferred path: not configured
    Default path: active
    Next hop: point2point
  Create time: 00:04:34, last status change time: 00:04:15
```

```
Signaling protocol: LDP, peer 10.1.1.1:0 up
  Targeted Hello: 10.1.1.1(LDP Id) -> 10.1.1.1
  MPLS VC labels: local 16, remote 17
  Group ID: local 0, remote 0
  MTU: local 1500, remote 1500
  Remote interface description:
  MAC Withdraw: sent 5, received 3
Sequencing: receive disabled, send disabled
VC statistics:
  packet totals: receive 0, send 0
  byte totals:   receive 0, send 0
  packet drops:  receive 0, send 0
```

Feature History for VPLS MAC Address Withdrawal

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Fuji 16.9.1	VPLS MAC Address Withdrawal	The VPLS MAC Address Withdrawal feature provides faster convergence by removing (or unlearning) MAC addresses that have been dynamically learned.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/

http://www.cisco.com/go/cfn.

CHAPTER 14

Configuring MPLS VPN Route Target Rewrite

Prerequisites for MPLS VPN Route Target Rewrite

- You should know how to configure Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs).
- You need to identify the RT replacement policy and target device for the autonomous system (AS).

Restrictions for MPLS VPN Route Target Rewrite

Route Target Rewrite can only be implemented in a single AS topology.

ip unnumbered command is not supported in MPLS configuration.

Information About MPLS VPN Route Target Rewrite

This section provides information about MPLS VPN Route Target Rewrite:

Route Target Replacement Policy

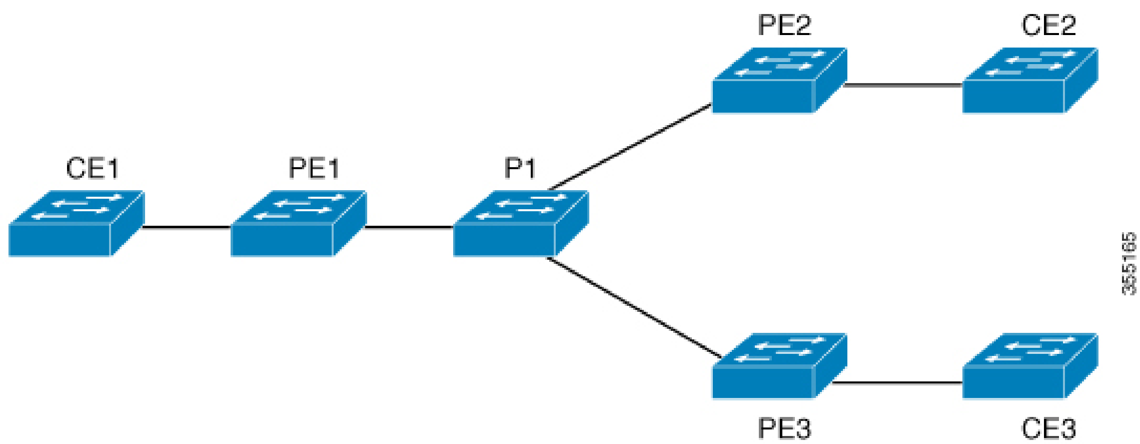
Routing policies for a peer include all configurations that may impact inbound or outbound routing table updates. The MPLS VPN Route Target Rewrite feature can influence routing table updates by allowing the replacement of route targets on inbound and outbound Border Gateway Protocol (BGP) updates. Route targets are carried as extended community attributes in BGP Virtual Private Network IP Version 4 (VPNv4) updates. Route target extended community attributes are used to identify a set of sites and VPN routing and forwarding (VRF) instances that can receive routes with a configured route target.

You can configure the MPLS VPN Route Target Rewrite feature on provider edge (PE) devices.

The figure below shows an example of route target replacement on PE devices in an Multiprotocol Label Switching (MPLS) VPN single autonomous system topology. This example includes the following configurations:

- PE1 is configured to import and export RT 65000:1 for VRF Customer A and to rewrite all inbound VPNv4 prefixes with RT 65000:1 to RT 65000:2.
- PE2 is configured to import and export RT 65000:2 for VRF Customer B and to rewrite all inbound VPNv4 prefixes with RT 65000:2 to RT 65000:1.

Figure 17: Route Target Replacement on Provide Edge(PE) devices in a single MPLS VPN Autonomous System Topology



Route Maps and Route Target Replacement

The MPLS VPN Route Target Rewrite feature extends the Border Gateway Protocol (BGP) inbound/outbound route map functionality to enable route target replacement. The **set extcomm-list delete** command entered in route-map configuration mode allows the deletion of a route target extended community attribute based on an extended community list.

How to Configure MPLS VPN Route Target Rewrite

This section provides the configuration steps for MPLS VPN Route Target Rewrite:

Configuring a Route Target Replacement Policy

Perform this task to configure a route target (RT) replacement policy for your internetwork.

If you configure a provider edge (PE) device to rewrite RT *x* to RT *y* and the PE has a virtual routing and forwarding (VRF) instance that imports RT *x* , you need to configure the VRF to import RT *y* in addition to RT *x* .

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip extcommunity-list** {*standard-list-number* | *expanded-list-number*} {**permit** | **deny**} [*regular-expression*] [**rt** | **soo** *extended-community-value*] **Example:** `Device(config)# ip extcommunity-list 1 permit rt 65000:2`	Creates an extended community access list and controls access to it. • The *standard-list-number* argument is an integer from 1 to 99 that identifies one or more permit or deny groups of extended communities. • The *expanded-list-number* argument is an integer from 100 to 500 that identifies one or more permit or deny groups of extended communities. Regular expressions can be configured with expanded lists but not standard lists. • The **permit** keyword permits access for a matching condition. • The **deny** keyword denies access for a matching condition. • The *regular-expression* argument specifies an input string pattern to match against. When you use an expanded extended community list to match route targets, include the pattern RT: in the regular expression. • The **rt** keyword specifies the route target extended community attribute. The **rt** keyword can be configured only with standard extended community lists and not expanded community lists. • The **soo** keyword specifies the site of origin (SOO) extended community attribute. The **soo** keyword can be configured only with standard extended community lists and not expanded community lists.

	Command or Action	Purpose
		• The *extended-community-value* argument specifies the route target or site of origin. The value can be one of the following combinations: • autonomous-system-number:network-number • ip-address:network-number The colon is used to separate the autonomous system number and network number or IP address and network number.
Step 4	**route-map** *map-name* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map rtrewrite permit 10`	Defines the conditions for redistributing routes from one routing protocol into another or enables policy routing and enables route-map configuration mode. • The *map-name* argument defines a meaningful name for the route map. The **redistribute** router configuration command uses this name to reference this route map. Multiple route maps can share the same map name. • If the match criteria are met for this route map, and the **permit** keyword is specified, the route is redistributed as controlled by the set actions. In the case of policy routing, the packet is policy routed. If the match criteria are not met, and the **permit** keyword is specified, the next route map with the same map tag is tested. If a route passes none of the match criteria for the set of route maps sharing the same name, it is not redistributed by that set. The **permit** keyword is the default. • If the match criteria are met for the route map and the **deny** keyword is specified, the route is not redistributed. In the case of policy routing, the packet is not policy routed, and no further route maps sharing the same map tag name will be examined. If the packet is not policy routed, the normal forwarding algorithm is used. • The *sequence-number* argument is a number that indicates the position a new route map will have in the list of route maps already configured with the same

	Command or Action	Purpose
		name. If given with the **no** form of this command, the position of the route map should be deleted.

Step 5	**match extcommunity** {*standard-list-number* | *expanded-list-number*}
	Example:
	`Device(config-route-map)# match extcommunity 1`
	Example:
	`Device(config-route-map)# match extcommunity 101`
		Matches the Border Gateway Protocol (BGP) extended community list attributes.
		• The *standard-list-number* argument is a number from 1 to 99 that identifies one or more permit or deny groups of extended community attributes.
		• The *expanded-list-number* argument is a number from 100 to 500 that identifies one or more permit or deny groups of extended community attributes.

Step 6	**set extcomm-list** *extended-community-list-number* **delete** **Example:** `Device(config-route-map)# set extcomm-list 1 delete`	Removes a route target from an extended community attribute of an inbound or outbound BGP Virtual Private Network Version 4 (VPNv4) update. • The *extended-community-list-number* argument specifies the extended community list number.

Step 7	**set extcommunity** {**rt** *extended-community-value* [**additive**] | **soo** *extended-community-value*}
	Example:
	`Device(config-route-map)# set extcommunity rt 65000:1 additive`
		Sets BGP extended community attributes.
		• The **rt** keyword specifies the route target extended community attribute.
		• The **soo** keyword specifies the site of origin extended community attribute.
		• The *extended-community-value* argument specifies the value to be set. The value can be one of the following combinations:
		  • autonomous-system-number : network-number
		  • ip-address : network-number
		The colon is used to separate the autonomous system number and network number or IP address and network number.
		• The **additive** keyword adds a route target to the existing route target list without replacing any existing route targets.

	Command or Action	Purpose
Step 8	**end** **Example:** `Device(config-route-map)# end`	(Optional) Returns to privileged EXEC mode.
Step 9	**show route-map** *map-name* **Example:** `Device# show route-map extmap`	(Optional) Verifies that the match and set entries are correct. • The *map-name* argument is the name of a specific route map.

Applying the Route Target Replacement Policy

Perform the following tasks to apply the route target replacement policy to your network:

Associating Route Maps with Specific BGP Neighbors

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Configures a Border Gateway Protocol (BGP) routing process and places the device in router configuration mode. • The *as-number* argument indicates the number of an autonomous system that identifies the device to other BGP devices and tags the routing information passed along. The range is 0 to 65535. Private autonomous system numbers that can be used in internal networks range from 64512 to 65535.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *as-number* **Example:**	Adds an entry to the BGP or multiprotocol BGP neighbor table.

	Command or Action	Purpose
	`Device(config-router)# neighbor 172.10.0.2 remote-as 200`	• The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group. • The *as-number* argument specifies the autonomous system to which the neighbor belongs.
Step 5	**address-family vpnv4** [**unicast**] **Example:** `Device(config-router)# address-family vpnv4`	Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard Virtual Private Network Version 4 (VPNv4) address prefixes. • The optional **unicast** keyword specifies VPNv4 unicast address prefixes.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 172.16.0.2 activate`	Enables the exchange of information with a neighboring BGP device. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **send-community** [**both** | **extended** | **standard**] **Example:** `Device(config-router-af)# neighbor 172.16.0.2 send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor. • The *ip-address* argument specifies the IP address of the BGP-speaking neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group. • The **both** keyword sends standard and extended community attributes. • The **extended** keyword sends an extended community attribute. • The **standard** keyword sends a standard community attribute.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name* {**in** | **out**} **Example:** `Device(config-router-af)# neighbor 172.16.0.2 route-map extmap in`	Apply a route map to incoming or outgoing routes • The *ip-address* argument specifies the IP address of the neighbor.

	Command or Action	Purpose
		• The *peer-group-name* argument specifies the name of a BGP or multiprotocol peer group. • The *map-name* argument specifies the name of a route map. • The **in** keyword applies route map to incoming routes. • The **out** keyword applies route map to outgoing routes.
Step 9	**end** **Example:** `Device(config-router-af)# end`	(Optional) Returns to privileged EXEC mode.

Verifying the Route Target Replacement Policy

Procedure

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

Example:

```
Device> enable
Device#
```

Step 2 **show ip bgp vpnv4 vrf** *vrf-name*

Verifies that Virtual Private Network Version 4 (VPNv4) prefixes with a specified route target (RT) extended community attribute are replaced with the proper RT extended community attribute to verify that the provider edge (PE) devices receive the rewritten RT extended community attributes.

Verify route target replacement on PE1:

Example:

```
Device# show ip bgp vpnv4 vrf Customer_A 192.168.1.1/32 internal
BGP routing table entry for 65000:1:192.168.1.1/32, version 6901
Paths: (1 available, best #1, table Customer_A)
  Advertised to update-groups:
     5
  Refresh Epoch 1
  650002
    3.3.3.3 (metric 3) (via default) from 3.3.3.3 (55.5.4.1)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Extended Community: RT:65000:1
      mpls labels in/out nolabel/3025
```

```
rx pathid: 0, tx pathid: 0x0
net: 0xFFB0A72E38, path: 0xFFB0E6A370, pathext: 0xFFB0E5D970
flags: net: 0x0, path: 0x7, pathext: 0x181
```

Step 3 **exit**

Returns to user EXEC mode:

Example:

```
Device# exit
Device>
```

Configuration Examples for MPLS VPN Route Target Rewrite

The following section provides configuration examples for MPLS VPN Route Target Rewrite:

Examples: Applying Route Target Replacement Policies

Examples: Associating Route Maps with Specific BGP Neighbor

This example shows the association of route map extmap with a Border Gateway Protocol (BGP) neighbor. The BGP inbound route map is configured to replace route targets (RTs) on incoming updates.

```
router bgp 1
address-family vpnv4
neighbor 2.2.2.2 route-map rtrewrite in
```

This example shows the association of the same route map with the outbound BGP neighbor. The route map is configured to replace RTs on outgoing updates.

```
router bgp 1
address-family vpnv4
neighbor 2.2.2.2 route-map rtrewrite out
```

Feature History for MPLS VPN Route Target Rewrite

This table provides release and related information for features explained in this module.

These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	MPLS VPN Route Target Rewrite	The MPLS VPN Route Target Rewrite feature can influence routing table updates by allowing the replacement of route targets on inbound and outbound Border Gateway Protocol (BGP) updates.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

CHAPTER 15

Configuring MPLS VPN-Inter-AS-IPv4 BGP Label Distribution

MPLS VPN Inter-AS IPv4 BGP Label Distribution

This feature enables you to set up a Virtual Private Network (VPN) service provider network. In this network, the Autonomous System Boundary Routers (ASBRs) exchange IPv4 routes with Multiprotocol Label Switching (MPLS) labels of the provider edge (PE) routers. Route reflectors (RRs) exchange VPNv4 routes by using multihop, multiprotocol, External Border Gateway Protocol (EBGP). This configuration saves the ASBRs from having to store all the VPNv4 routes. Using the route reflectors to store the VPNv4 routes and forward them to the PE routers results in improved scalability.

The MPLS VPN—Inter-AS—IPv4 BGP Label Distribution feature has the following benefits:

- Having the route reflectors store VPNv4 routes results in improved scalability—This configuration scales better than configurations where the ASBR holds all the VPNv4 routes and forwards the routes based on VPNv4 labels. With this configuration, route reflectors hold the VPNv4 route, which 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 LSRs—If two adjacent label switch routers (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.
- Includes EBGP multipath support to enable load balancing for IPv4 routes across autonomous system (AS) boundaries.

Restrictions for MPLS VPN Inter-AS IPv4 BGP Label Distribution

This feature includes the following restrictions:

- For networks configured with EBGP multihop, a labeled switched path (LSP) must be established between nonadjacent devices. (RFC 3107)
- The PE devices must run images that support BGP label distribution. Otherwise, you cannot run EBGP between them.
- Point-to-Point Protocol (PPP) encapsulation on the ASBRs is not supported with this feature.
- The physical interfaces that connect the BGP speakers must support Cisco Express Forwarding (CEF) or distributed CEF and MPLS

Information About MPLS VPN Inter-AS IPv4 BGP Label Distribution

To configure MPLS VPN Inter-AS IPv4 BGP Label Distribution, you need the following information:

MPLS VPN Inter-AS IPv4 BGP Label Distribution Overview

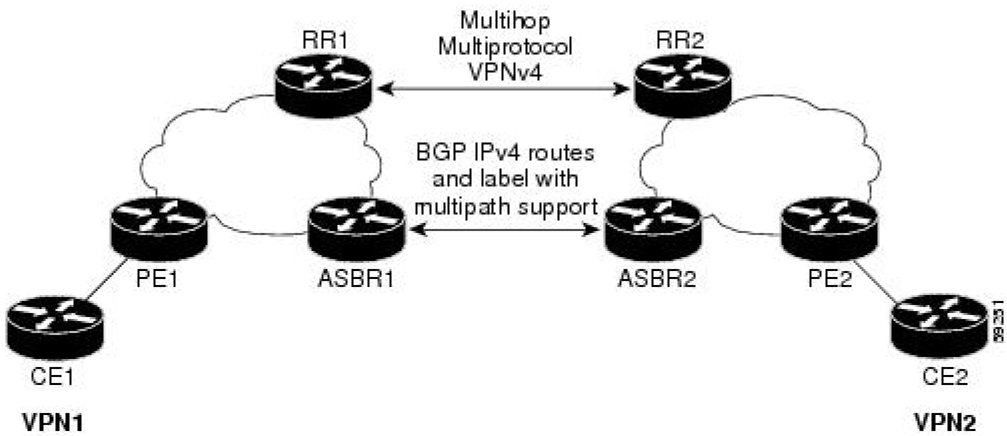
This feature enables you to set up a VPN service provider network to exchange IPv4 routes with MPLS labels. You can configure the VPN service provider network as follows:

- Route reflectors exchange VPNv4 routes by using multihop, multiprotocol EBGP. This configuration also preserves the next hop information and the VPN labels across the autonomous systems.
- A local PE router (for example, PE1 in Figure 1) 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 vice versa.
  - Internal Border Gateway Protocol (IBGP) IPv4 label distribution: The ASBR and PE router can use direct IBGP sessions to exchange VPNv4 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 is accomplished by enabling the ASBR to exchange IPv4 routes and MPLS labels with the route reflector. The route reflector also reflects the VPNv4 routes to the PE routers in the VPN (as mentioned in the first bullet). For example, in VPN1, RR1 reflects to PE1 the VPNv4 routes it learned and IPv4 routes and MPLS labels learned from ASBR1. Using the route reflectors to store the VPNv4 routes and forward them through the PE routers and ASBRs allows for a scalable configuration.

- ASBRs exchange IPv4 routes and MPLS labels for the PE routers by using EBGP. This enables load balancing across CSC boundaries.

Figure 18: VPNs Using EBGP and IBGP to Distribute Routes and MPLS Labels



BGP Routing Information

BGP routing information includes the following items:

- A 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 its way to the local router. The first autonomous system in the list is closest to the local router. The last autonomous system in the list is farthest from the local router and usually the autonomous system where the route began.
- Path attributes, which provide other information about the autonomous system path, for example, the next hop.

How BGP Sends 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 **neighbor send-label** 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.

Using Route Maps to Filter Routes

When both routers are configured to distribute routes with MPLS labels, all the routes are encoded with the multiprotocol extensions and contain an MPLS label. You can use a route map to control the distribution of MPLS labels between routers. Route maps enable you to specify the following:

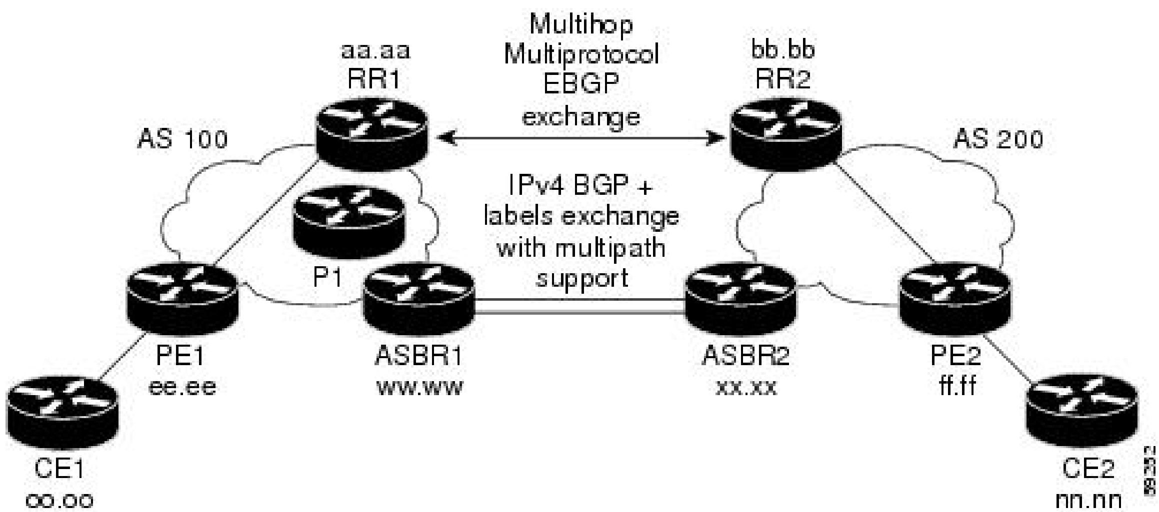
- For a router distributing MPLS labels, you can specify which routes are distributed with an MPLS label.
- For a router receiving MPLS labels, you can specify which routes are accepted and installed in the BGP table.

How to Configure MPLS VPN Inter-AS IPv4 BGP Label Distribution

The figure below shows the following configuration:

- The configuration consists of two VPNs.
- The ASBRs exchange the IPv4 routes with MPLS labels.
- The route reflectors exchange the VPNv4 routes using multi-hop MPLS EBGP.
- The route reflectors reflect the IPv4 and VPNv4 routes to the other routers in its autonomous system.

Figure 19: Configuring Two VPN Service Providers to Exchange IPv4 Routes and MPLS Labels



Configuring the ASBRs to Exchange IPv4 Routes and MPLS Labels

Perform this task to configure the ASBRs so that they can distribute BGP routes with MPLS labels.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Enters router configuration mode. • as-number—Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. The valid values range from 1 through 65535. Private autonomous system numbers that can be used in internal networks range from 64512 through 65535.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *as-number* **Example:** `Device(config)# neighbor 209.165.201.2 remote-as 200`	Adds an entry to the BGP or multiprotocol BGP neighbor table. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name*argument specifies the name of a BGP peer group. • The *as-number* argument specifies the autonomous system to which the neighbor belongs.
Step 5	**address-family ipv4** [**multicast** | **unicast** | **vrf***vrf-name*] **Example:** `Device(config-router)# address-family ipv4`	Enters address family configuration mode for configuring routing sessions such as BGP that use standard IPv4 address prefixes. • The **multicast**keyword specifies IPv4 multicast address prefixes. • The **unicast**keyword specifies IPv4 unicast address prefixes. • The vrf vrf-name keyword and argument specifies the name of the VPN routing/forwarding instance (VRF) to associate with subsequent IPv4 address family configuration mode commands.
Step 6	**maximum-paths** *number-paths* **Example:** `Device(config-router)# maximum-paths 2`	(Optional) Controls the maximum number of parallel routes an IP routing protocol can support. The number-paths argument specifies the maximum number of parallel routes an IP routing protocol installs in a routing table, in the range from 1 through 6.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:**	Enables the exchange of information with a neighboring router.

	Command or Action	Purpose
	`Device(config-router-af)# neighbor 209.165.201.2 activate`	• The ip-address argument specifies the IP address of the neighbor. • The peer-group-name argument specifies the name of a BGP peer group.
Step 8	**neighbor** *ip-address***send-label** **Example:** `Device(config-router-af)# neighbor 10.0.0.1 send-label`	Enables a BGP router to send MPLS labels with BGP routes to a neighboring BGP router. • The ip-address argument specifies the IP address of the neighboring router.
Step 9	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits from the address family submode.
Step 10	**end** **Example:** `Device(config-router-af)# end`	(Optional) Exits to privileged EXEC mode.

Configuring the Route Reflectors to Exchange VPNv4 Routes

Before you begin

Perform this task to enable the route reflectors to exchange VPNv4 routes by using multihop, multiprotocol EBGP.

This procedure also specifies that the next hop information and the VPN label are preserved across the autonomous systems. This procedure uses RR1 as an example.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Enters router configuration mode. • as-number—Number of an autonomous system that identifies the router to other

	Command or Action	Purpose
		BGP routers and tags the routing information that is passed along. The valid values range from 1through 65535. Private autonomous system numbers that can be used in internal networks range from 64512 through 65535. The autonomous system number identifies RR1 to routers in other autonomous systems.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *as-number* **Example:** `Device(config)# neighbor 192.0.2.1 remote-as 200`	Adds an entry to the BGP or multiprotocol BGP neighbor table. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name*argument specifies the name of a BGP peer group. • The *as-number* argument specifies the autonomous system to which the neighbor belongs.
Step 5	**address-family vpnv4** [**unicast**] **Example:** `Device(config-router)# address-family vpnv4`	Enters address family configuration mode for configuring routing sessions, such as BGP, that uses standard Virtual Private Network Version 4 (VPNv4) address prefixes. • The optional **unicast** keyword specifies VPNv4 unicast address prefixes.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **ebgp-multihop** [*ttl*] **Example:** `Device(config-router-af)# neighbor 192.0.2.1 ebgp-multihop 255`	Accepts and attempts BGP connections to external peers residing on networks that are not directly connected. • The *ip-address* argument specifies the IP address of the BGP-speaking neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group. • The*ttl* argument specifies the time-to-live in the range from 1 through 255 hops.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 192.0.2.1 activate`	Enables the exchange of information with a neighboring router. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group.

	Command or Action	Purpose
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **next-hop unchanged** **Example:** `Device(config-router-af)# neighbor 10.0.0.2 next-hop unchanged`	Enables an External BGP (EBGP) multihop peer to propagate the next hop unchanged. • The *ip-address* argument specifies the IP address of the next hop. • The *peer-group-name* argument specifies the name of a BGP peer group that is the next hop.
Step 9	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits from the address family submode.
Step 10	**end** **Example:** `Device(config-router-af)# end`	(Optional) Exits to privileged EXEC mode.

Configuring the Route Reflectors to Reflect Remote Routes in Its autonomous system

Perform this task to enable the RR to reflect the IPv4 routes and labels that are learned by the ASBR to the PE routers in the autonomous system.

This is accomplished by making the ASBR and PE router the route reflector clients of the RR. This procedure also explains how to enable the RR to reflect the VPNv4 routes.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Enters router configuration mode. • as-number—Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. The

	Command or Action	Purpose
		valid values range from 1 through 65535. Private autonomous system numbers that can be used in internal networks range from 64512 through 65535. The autonomous system number identifies RR1 to routers in other autonomous systems.
Step 4	**address-family ipv4** [**multicast** | **unicast** | **vrf***vrf-name*] **Example:** `Device(config-router)# address-family ipv4`	Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard IPv4 address prefixes. • The **multicast**keyword specifies IPv4 multicast address prefixes. • The **unicast**keyword specifies IPv4 unicast address prefixes. • The vrf vrf-name keyword and argument specifies the name of the VPN routing/forwarding instance (VRF) to associate with subsequent IPv4 address family configuration mode commands.
Step 5	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 203.0.113.1 activate`	Enables the exchange of information with a neighboring router. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group.
Step 6	**neighbor***ip-address***route-reflector-client** **Example:** `Device(config-router-af)# neighbor 203.0.113.1 route-reflector-client`	Configures the router as a BGP route reflector and configures the specified neighbor as its client. • The ip-address argument specifies the IP address of the BGP neighbor being identified as a client.
Step 7	**neighbor***ip-address***send-label** **Example:** `Device(config-router-af)# neighbor 203.0.113.1 send-label`	Enables a BGP router to send MPLS labels with BGP routes to a neighboring BGP router. • The ip-address argument specifies the IP address of the neighboring router.
Step 8	**exit-address-family** **Example:**	Exits from the address family submode.

	Command or Action	Purpose
	`Device(config-router-af)# exit-address-family`	
Step 9	**address-family vpnv4** [**unicast**] **Example:** `Device(config-router)# address-family vpnv4`	Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes. • The optional **unicast** keyword specifies VPNv4 unicast address prefixes.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 203.0.113.1 activate`	Enables the exchange of information with a neighboring router. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group.
Step 11	**neighbor** *ip-address* **route-reflector-client** **Example:** `Device(config-router-af)# neighbor 203.0.113.1 route-reflector-client`	Enables the RR to pass IBGP routes to the neighboring router.
Step 12	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits from the address family submode.
Step 13	**end** **Example:** `Device(config-router-af)# end`	(Optional) Exits to privileged EXEC mode.

Creating Route Maps

Route maps enable you to specify which routes are distributed with MPLS labels. Route maps also enable you to specify which routes with MPLS labels a router receives and adds to its BGP table.

Route maps work with access lists. You enter the routes into an access list and then specify the access list when you configure the route map.

The following procedures enable the ASBRs to send MPLS labels with the routes specified in the route maps. Further, the ASBRs accept only the routes that are specified in the route map.

Configuring a Route Map for Arriving Routes

Perform this task to create a route map to filter arriving routes. You create an access list and specify the routes that the router accepts and adds to the BGP table.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Enters router configuration mode. • as-number—Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. The valid values range from 1 through 65535. Private autonomous system numbers that can be used in internal networks range from 64512 through 65535. The autonomous system number identifies RR1 to routers in other autonomous systems.
Step 4	**route-map** *route-map name* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config-router)# route-map IN permit 11`	Creates a route map with the name you specify. • The **permit** keyword allows the actions to happen if all conditions are met. • The **deny**keyword prevents any actions from happening if all conditions are met. • The *sequence-number* argument allows you to prioritize route maps. If you have multiple route maps and want to prioritize them, assign each one a number. The route map with the lowest number is implemented first, followed by the route map with the second lowest number, and so on.
Step 5	**match ip address** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:** `Device(config-route-map)# match ip address 2`	Distributes any routes that have a destination network number address that is permitted by a standard or extended access list, or performs policy routing on packets. • The *access-list-number* argument is a number of a standard or extended access

	Command or Action	Purpose
		list. It can be an integer from 1 through 199. • The *access-list-name* argument is a name of a standard or extended access list. It can be an integer from 1 through 199.
Step 6	**match mpls-label** **Example:** `Device(config-route-map)# match mpls-label`	Redistributes routes that include MPLS labels if the routes meet the conditions that are specified in the route map.
Step 7	**end** **Example:** `Device(config-router-af)# end`	(Optional) Exits to privileged EXEC mode.

Configuring a Route Map for Departing Routes

Perform this task to create a route map to filter departing routes. You create an access list and specify the routes that the router distributes with MPLS labels.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Enters router configuration mode. • as-number—Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. The valid values range from 1 through 65535. Private autonomous system numbers that can be used in internal networks range from 64512 through 65535. The AS number identifies RR1 to routers in other autonomous systems.

	Command or Action	Purpose
Step 4	**route-map** *route-map name* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config-router)# route-map OUT permit 10`	Creates a route map with the name you specify. • The **permit** keyword allows the actions to happen if all conditions are met. • The **deny**keyword prevents any actions from happening if all conditions are met. • The *sequence-number* argument allows you to prioritize route maps. If you have multiple route maps and want to prioritize them, assign each one a number. The route map with the lowest number is implemented first, followed by the route map with the second lowest number, and so on.
Step 5	**match ip address** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:** `Device(config-route-map)# match 10.0.0.2 1`	Distributes any routes that have a destination network number address that is permitted by a standard or extended access list, or performs policy routing on packets. • The *access-list-number* argument is a number of a standard or extended access list. It can be an integer from 1 through 199. • The *access-list-name* argument is a name of a standard or extended access list. It can be an integer from 1 through 199.
Step 6	**set mpls-label** **Example:** `Device(config-route-map)# set mpls-label`	Enables a route to be distributed with an MPLS label if the route matches the conditions that are specified in the route map.
Step 7	**end** **Example:** `Device(config-router-af)# end`	(Optional) Exits to privileged EXEC mode.

Applying the Route Maps to the ASBRs

Perform this task to enable the ASBRs to use the route maps.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

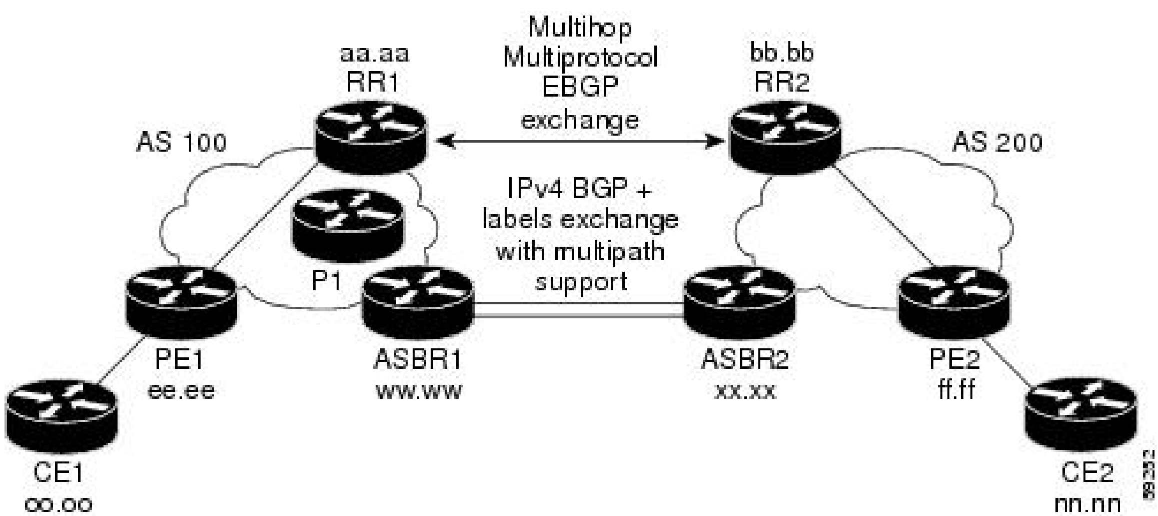
	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# router bgp 100`	Enters router configuration mode. • as-number—Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. The valid values range from 1 through 65535. Private autonomous system numbers that can be used in internal networks range from 64512 through 65535. The autonomous system number identifies RR1 to routers in other autonomous systems.
Step 4	**address-family ipv4** [**multicast** | **unicast** | **vrf***vrf-name*] **Example:** `Device(config-router)# address-family ipv4`	Enters address family configuration mode for configuring routing sessions such as BGP that use standard IPv4 address prefixes. • The **multicast**keyword specifies IPv4 multicast address prefixes. • The **unicast**keyword specifies IPv4 unicast address prefixes. • The vrf vrf-name keyword and argument specifies the name of the VPN routing/forwarding instance (VRF) to associate with subsequent IPv4 address family configuration mode commands.
Step 5	**neighbor***ip-address***route-map***route-map-name***out** **Example:** `Device(config-router-af)# neighbor 209.165.200.225 route-map OUT out`	Applies a route map to incoming routes. • The *ip-address* argument specifies the device to which the route map is to be applied. • The *route-map-name* argument specifies the name of the route map. • The **out** keyword applies the route map to outgoing routes.

	Command or Action	Purpose
Step 6	**neighbor** *ip-address* **send-label** **Example:** `Device(config-router-af)# neighbor 209.165.200.225 send-label`	Advertises the ability of the router to send MPLS labels with routes. • The ip-address argument specifies the router that is enabled to send MPLS labels with routes.
Step 7	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits from the address family submode.
Step 8	**end** **Example:** `Device(config-router-af)# end`	(Optional) Exits to privileged EXEC mode.

Verifying the MPLS VPN Inter-AS IPv4 BGP Label Distribution Configuration

The following figure is a reference for the configuration.

Figure 20: Configuring Two VPN Service Providers to Exchange IPv4 Routes and MPLS Labels



If you use route reflectors to distribute the VPNv4 routes and use the ASBRs to distribute the IPv4 labels, use the following procedures to help verify the configuration:

Verifying the Route Reflector Configuration

Perform this task to verify the route reflector configuration.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	**show ip bgp vpnv4** {**all** | **rd** *route-distinguisher* | **vrf** *vrf-name*} [**summary**] [**labels**]
	Example:
	`Device# show ip bgp vpnv4 all summary`
	Example:
	`Device# show ip bgp vpnv4 all labels`
		(Optional) Displays VPN address information from the BGP table.
		• Use the **show ip bgp vpnv4** command with the all and summary keywords to verify that a multihop, multiprotocol, EBGP session exists between the route reflectors and that the VPNv4 routes are being exchanged between the route reflectors.
		• The last two lines of the command output show the following information:
		  • Prefixes are being learned from PE1 and then passed to RR2.
		  • Prefixes are being learned from RR2 and then passed to PE1.
		• Use the **show ip bgp vpnv4** command with the all and labels keywords to verify that the route reflectors are exchanging VPNv4 label information.

Step 3	**disable** **Example:** `Device# disable`	(Optional) Exits to user EXEC mode.

Verifying that CE1 Has Network Reachability Information for CE2

Perform this task to verify that router CE1 has NLRI for router CE2.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**show ip route** [*ip-address* [*mask*] [**longer prefixes**]] | [*protocol* [*process-id*]] | [**list** *access-list-number* | *access-list-name*] **Example:** `Device# show ip route 209.165.201.1`	Displays the current state of the routing table. • Use the **show ip route** command with the ip-address argument to verify that CE1 has a route to CE2. • Use the **show ip route** command to verify the routes learned by CE1. Make sure to list the route for CE2.
Step 3	**disable** **Example:** `Device# disable`	(Optional) Exits to user EXEC mode.

Verifying that PE1 Has Network Layer Reachability Information for CE2

Perform this task to verify that router PE1 has NLRI for router CE2.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip route vrf** *vrf-name* [**connected**] [*protocols* [*as-number*] [*tag*] [*output-modifiers*]] [**list** *number* [*output-modifiers*]] [**profile**] [**static** [*output-modifiers*]] [**summary** [*output-modifiers*]] [**supernets-only** [*output-modifiers*]] [**traffic engineering** [*output-modifiers*]] **Example:** `Device# show ip route vrf vpn1 209.165.201.1`	(Optional) Displays the IP routing table that is associated with a VRF. • Use the **show ip route vrf** command to verify that router PE1 learns routes from router CE2 (nn.nn.nn.nn).
Step 3	**show ip bgp vpnv4** {**all** | **rd** *route-distinguisher* | **vrf** *vrf-name*} {*ip-prefix/length* [**longer-prefixes**] [*output-modifiers*]] [*network-address* [*mask*] [**longer-prefixes**] [*output-modifiers*]] [**cidr-only**] [*community*] [**community-list**] [**dampened-paths**] [**filter-list**] [**flap-statistics**] [**inconsistent-as**]	(Optional) Displays VPN address information from the BGP table. • Use the **show ip bgp vpnv4** command with the **vrf** or **all** keyword to verify that router PE2 is the BGP next-hop to router CE2.

	Command or Action	Purpose
	[**neighbors**] [**path** [*line*]] [**peer-group**] [**quote-regexp**] [**regexp**] [**summary**] [**tags**] **Example:** `Device# show ip bgp vpnv4 vrf vpn1 209.165.201.1`	
Step 4	**show ip cef** [**vrf** *vrf-name*] [*network* [*mask*]] [**longer-prefixes**] [**detail**] **Example:** `Device# show ip cef vrf vpn1 209.165.201.1`	(Optional) Displays entries in the forwarding information base (FIB) or displays a summary of the FIB. • Use the **show ip cef** command to verify that the Cisco Express Forwarding (CEF) entries are correct.
Step 5	**show mpls forwarding-table** [{*network* {*mask* | *length*} | **labels** *label* [*-label*] | **interface** *interface* | **next-hop** *address* | **lsp-tunnel** [*tunnel-id*] }] [**detail**] **Example:** `Device# show mpls forwarding-table`	(Optional) Displays the contents of the MPLS forwarding information base (LFIB). • Use the **show mpls forwarding-table** command to verify the IGP label for the BGP next hop router (autonomous system boundary).
Step 6	**show ip bgp** [*network*] [*network-mask*] [**longer-prefixes**] **Example:** `Device# show ip bgp 209.165.202.129`	(Optional) Displays entries in the BGP routing table. • Use the **show ip bgp** command to verify the label for the remote egress PE router (PE2).
Step 7	**show ip bgp vpnv4** {**all** | **rd** *route-distinguisher* | **vrf** *vrf-name*} [**summary**] [**labels**] **Example:** `Device# show ip bgp vpnv4 all labels`	(Optional) Displays VPN address information from the BGP table. • Use the **show ip bgp vpnv4 all summary** command to verify the VPN label of CE2, as advertised by PE2.
Step 8	**disable** **Example:** `Device# disable`	(Optional) Exits to user EXEC mode.

Verifying that PE2 Has Network Reachability Information for CE2

Perform this task to ensure that PE2 can access CE2.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ip route vrf** *vrf-name* [**connected**] [*protocol* [*as-number*] [*tag*] [*output-modifiers*]] [**list** *number* [*output-modifiers*]] [**profile**] [**static** [*output-modifiers*]] [**summary** [*output-modifiers*]] [**supernets-only** [*output-modifiers*]] [**traffic-engineering** [*output-modifiers*]] **Example:** `Device# show ip route vrf vpn1 209.165.201.1`	(Optional) Displays the IP routing table that is associated with a VRF. • Use the show ip route vrf command to check the VPN routing and forwarding table for CE2. The output provides next hop information.
Step 3	**show mpls forwarding-table** [**vrf** *vpn-name*] [{*network* {*mask* | *length* } | **labels** *label* [*-label*] | **interface** *interface* | **next-hop** *address* | **lsp-tunnel** [*tunnel-id*] }] [**detail**] **Example:** `Device# show mpls forwarding-table vrf vpn1 209.165.201.1`	(Optional) Displays the contents of the LFIB. • Use the **show mpls forwarding-table** command with the **vrf** keyword to check the VPN routing and forwarding table for CE2. The output provides the label for CE2 and the outgoing interface.
Step 4	**show ip bgp vpnv4** {**all** | **rd** *route-distinguisher* | **vrf** *vrf-name*} [**summary**] [**labels**] **Example:** `Device# show ip bgp vpnv4 all labels`	(Optional) Displays VPN address information from the BGP table. • Use the **show ip bgp vpnv4** command with the **all** and **labels** keywords to check the VPN label for CE2 in the multiprotocol BGP table.
Step 5	**show ip cef** [**vrf** *vrf-name*] [*network* [*mask*]] [**longer-prefixes**] [**detail**] **Example:** `Device# show ip cef <vrf-name> 209.165.201.1`	(Optional) Displays entries in the forwarding information base (FIB) or displays a summary of the FIB. • Use the **show ip cef** command to check the CEF entry for CE2. The command output shows the local label for CE2 and the outgoing interface.
Step 6	**disable** **Example:** `Device# disable`	(Optional) Exits to user EXEC mode.

Verifying the ASBR Configuration

Perform this task to verify that the ASBRs exchange IPv4 routes with MPLS labels or IPv4 routes without labels as prescribed by a route map.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	**show ip bgp** [*network*] [*network-mask*] [**longer-prefixes**]
	Example:
	`Device# show ip bgp 209.165.202.129`
	Example:
	`Device# show ip bgp 192.0.2.1`
		(Optional) Displays entries in the BGP routing table.
		• Use the **show ip bgp** command to verify that
		  • ASBR1 receives an MPLS label for PE2 from ASBR2.
		  • ASBR1 received from ASBR2 IPv4 routes for RR2 without labels. If the command output does not display the MPLS label information, the route was received without an MPLS label.
		  • ASBR2 distributes an MPLS label for PE2 to ASBR1.
		  • ASBR2 does not distribute a label for RR2 to ASBR1.

Step 3	**show ip cef** [**vrf** *vrf-name*] [*network* [*mask*]] [**longer-prefixes**] [**detail**]
	Example:
	`Device# show ip cef 209.165.202.129`
	Example:
	`Device# show ip cef 192.0.2.1`
		(Optional) Displays entries in the forwarding information base (FIB) or displays a summary of the FIB.
		• Use the **show ip cef** command from ASBR1 and ASBR2 to check that
		  • The CEF entry for PE2 is correct.
		  • The CEF entry for RR2 is correct.

Step 4	**disable** **Example:** `Device# disable`	(Optional) Exits to the user EXEC mode.

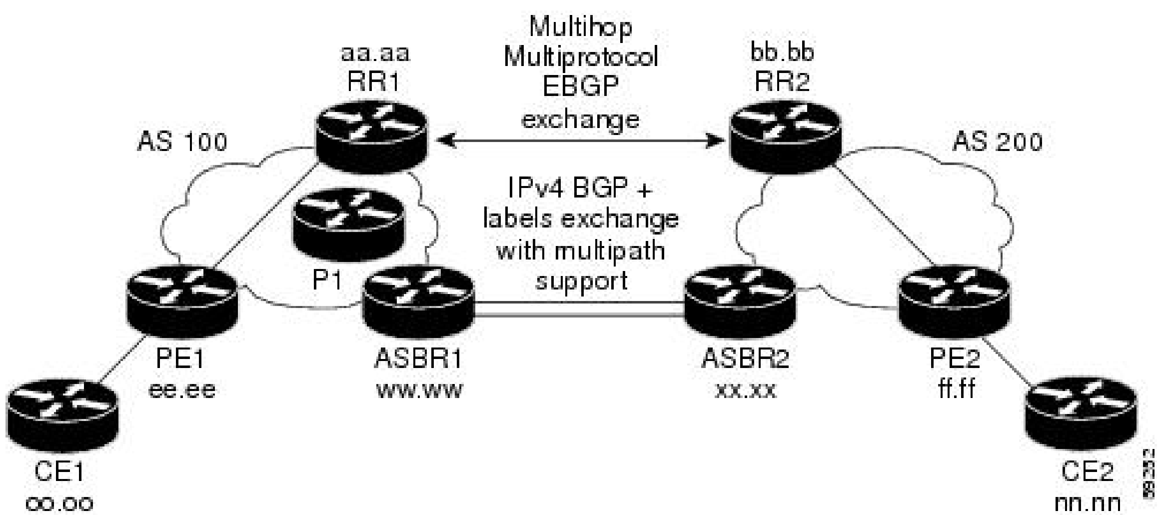
Configuration Examples for MPLS VPN Inter-AS IPv4 BGP Label Distribution

Configuration examples for MPLS VPN Inter-AS IPv4 BGP Label Distribution feature include the following:

Configuration Examples for Inter-AS Using BGP to Distribute Routes and MPLS Labels Over an MPLS VPN Service Provider

The figure shows two MPLS VPN service providers. The service provider distributes the VPNv4 routes between the route reflectors. They distribute the IPv4 routes with MPLS labels between the ASBRs.

Figure 21: Distributing IPv4 Routes and MPLS Labels Between MPLS VPN Service Providers



The configuration examples show the two techniques that you can use to distribute the VPNv4 routes and the IPv4 routes with MPLS labels, from the remote RRs and PEs to the local RRs and PEs:

- Autonomous system 100 uses the RRs to distribute the VPNv4 routes learned from the remote RRs. The RRs also distribute the remote PE address and label that is learned from ASBR1 using IPv4 + labels.
- In autonomous system 200, the IPv4 routes that ASBR2 learned are redistributed into IGP.

The configuration examples in this section are as follow:

Example: Route Reflector 1 (MPLS VPN Service Provider)

The configuration example for RR1 specifies the following:

- RR1 exchanges VPNv4 routes with RR2 using multiprotocol, multihop EBGP.
- The VPNv4 next hop information and the VPN label preserved across the autonomous systems.
- RR1 reflects to PE1:
    - The VPNv4 routes learned from RR2.

- The IPv4 routes and MPLS labels learned from ASBR1

```
ip subnet-zero
 ip cef
 !
 interface Loopback0
  ip address 10.0.0.1 255.255.255.255
  no ip directed-broadcast
 !
 interface Serial1/2
  ip address 209.165.201.8 255.0.0.0
  no ip directed-broadcast
  clockrate 124061
 !
 router ospf 10
  log-adjacency-changes
  auto-cost reference-bandwidth 1000
  network 10.0.0.1 0.0.0.0 area 100
  network 209.165.201.9 0.255.255.255 area 100
!
 router bgp 100
  bgp cluster-id 1
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 203.0.113.1 remote-as 100
  neighbor 203.0.113.1 update-source Loopback0
  neighbor 209.165.200.225 remote-as 100
  neighbor 209.165.200.225 update-source Loopback0
  neighbor 192.0.2.1 remote-as 200
  neighbor 192.0.2.1 ebgp-multihop 255
  neighbor 192.0.2.1 update-source Loopback0
  no auto-summary
  !
 address-family ipv4
  neighbor 203.0.113.1 activate
  neighbor 203.0.113.1 route-reflector-client                !IPv4+labels session to PE1
  neighbor 203.0.113.1 send-label
  neighbor 209.165.200.225 activate
  neighbor 209.165.200.225 route-reflector-client                !IPv4+labels session to
ASBR1
  neighbor 209.165.200.225 send-label
  no neighbor 192.0.2.1 activate
  no auto-summary
  no synchronization
  exit-address-family
  !
 address-family vpnv4
  neighbor 203.0.113.1 activate
  neighbor 203.0.113.1 route-reflector-client                !VPNv4 session with PE1
  neighbor 203.0.113.1 send-community extended
  neighbor 192.0.2.1 activate
  neighbor 192.0.2.1 next-hop-unchanged                     !MH-VPNv4 session with RR2
  neighbor 192.0.2.1 send-community extended                  !with next hop unchanged
  exit-address-family
 !
 ip default-gateway 3.3.0.1
 no ip classless
 !
 snmp-server engineID local 00000009020000D0584B25C0
 snmp-server community public RO
 snmp-server community write RW
 no snmp-server ifindex persist
 snmp-server packetsize 2048
```

```
 !
 end
```

Configuration Example: ASBR1 (MPLS VPN Service Provider)

ASBR1 exchanges IPv4 routes and MPLS labels with ASBR2.

In this example, ASBR1 uses route maps to filter routes.

- A route map called OUT specifies that ASBR1 should distribute the PE1 route (ee.ee) with labels and the RR1 route (aa.aa) without labels.
- A route map called IN specifies that ASBR1 should accept the PE2 route (ff.ff) with labels and the RR2 route (bb.bb) without labels.

```
ip subnet-zero
 mpls label protocol tdp
 !
 interface Loopback0
  ip address 209.165.200.225 255.255.255.255
  no ip directed-broadcast
  no ip route-cache
  no ip mroute-cache
 !
 interface Ethernet0/2
  ip address 209.165.201.6 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
 !
 interface Ethernet0/3
  ip address 209.165.201.18 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
  mpls label protocol ldp
  mpls ip
 !router ospf 10
  log-adjacency-changes
  auto-cost reference-bandwidth 1000
  redistribute connected subnets
  passive-interface Ethernet0/2
  network 209.165.200.225 0.0.0.0 area 100
  network 209.165.201.9 0.255.255.255 area 100

 router bgp 100
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 10.0.0.1 remote-as 100
  neighbor 10.0.0.1 update-source Loopback0
  neighbor 209.165.201.2 remote-as 200
  no auto-summary
  !
 address-family ipv4                          ! Redistributing IGP into BGP
  redistribute ospf 10                        ! so that PE1 & RR1 loopbacks
  neighbor 10.0.0.1 activate            ! get into the BGP table
  neighbor 10.0.0.1 send-label
  neighbor 209.165.201.2 activate
  neighbor 209.165.201.2 advertisement-interval 5
  neighbor 209.165.201.2 send-label
  neighbor 209.165.201.2 route-map IN in        ! accepting routes in route map IN.
  neighbor 209.165.201.2 route-map OUT out      ! distributing routes in route map OUT.
  neighbor 209.165.201.3 activate
  neighbor 209.165.201.3 advertisement-interval 5
  neighbor 209.165.201.3 send-label
```

```
  neighbor 209.165.201.3 route-map IN in        ! accepting routes in route map IN.
  neighbor 209.165.201.3 route-map OUT out      ! distributing routes in route map OUT.
  no auto-summary
  no synchronization
  exit-address-family
 !
 ip default-gateway 3.3.0.1
 ip classless
 !
 access-list 1 permit 203.0.113.1 log                !Setting up the access lists
 access-list 2 permit 209.165.202.129 log
 access-list 3 permit 10.0.0.1 log
 access-list 4 permit 192.0.2.1 log

 route-map IN permit 10                              !Setting up the route maps
  match ip address 2
  match mpls-label
 !
 route-map IN permit 11
  match ip address 4
 !
 route-map OUT permit 12
  match ip address 3
 !
 route-map OUT permit 13
  match ip address 1
  set mpls-label
 !
 end
```

Configuration Example: Route Reflector 2 (MPLS VPN Service Provider)

RR2 exchanges VPNv4 routes with RR1 through multihop, multiprotocol EBGP. This configuration also specifies that the next hop information and the VPN label are preserved across the autonomous systems.

```
ip subnet-zero
 ip cef
 !
 interface Loopback0
  ip address 192.0.2.1 255.255.255.255
  no ip directed-broadcast
 !
 interface Serial1/1
  ip address 209.165.201.10 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
 !
 router ospf 20
  log-adjacency-changes
  network 192.0.2.1 0.0.0.0 area 200
  network 209.165.201.20 0.255.255.255 area 200
 !
router bgp 200
  bgp cluster-id 1
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 10.0.0.1 remote-as 100
  neighbor 10.0.0.1 ebgp-multihop 255
  neighbor 10.0.0.1 update-source Loopback0
  neighbor 209.165.202.129 remote-as 200
  neighbor 209.165.202.129 update-source Loopback0
  no auto-summary
  !
  address-family vpnv4
```

```
  neighbor 10.0.0.1 activate
  neighbor 10.0.0.1 next-hop-unchanged              !Multihop VPNv4 session with RR1
  neighbor 10.0.0.1 send-community extended              !with next-hop-unchanged
  neighbor 209.165.202.129 activate
  neighbor 209.165.202.129 route-reflector-client          !VPNv4 session with PE2
  neighbor 209.165.202.129 send-community extended
  exit-address-family
 !
 ip default-gateway 3.3.0.1
 no ip classless
 !
 end
```

Configuration Example: ASBR2 (MPLS VPN Service Provider)

ASBR2 exchanges IPv4 routes and MPLS labels with ASBR1. However, in contrast to ASBR1, ASBR2 does not use the RR to reflect IPv4 routes and MPLS labels to PE2. ASBR2 redistributes the IPv4 routes and MPLS labels learned from ASBR1 into IGP. PE2 can now reach these prefixes.

```
ip subnet-zero
 ip cef
 !
 mpls label protocol tdp
 !
 interface Loopback0
  ip address 209.165.200.226 255.255.255.255
  no ip directed-broadcast
 !
 interface Ethernet1/0
  ip address 209.165.201.2 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
 !
 interface Ethernet1/2
  ip address 209.165.201.4 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
  mpls label protocol tdp
  mpls ip
  !
router ospf 20
  log-adjacency-changes
  auto-cost reference-bandwidth 1000
  redistribute connected subnets
  redistribute bgp 200 subnets           ! Redistributing the routes learned from
  passive-interface Ethernet1/0               ! ASBR1(EBGP+labels session) into IGP
  network 209.165.200.226 0.0.0.0 area 200         ! so that PE2 will learn them
  network 209.165.201.5 0.255.255.255 area 200
  !
 router bgp 200
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 192.0.2.1 remote-as 200
  neighbor 192.0.2.1 update-source Loopback0
  neighbor 209.165.201.6 remote-as 100
  no auto-summary
  !
address-family ipv4
  redistribute ospf 20                         ! Redistributing IGP into BGP
  neighbor 209.165.201.6 activate                   ! so that PE2 & RR2 loopbacks
  neighbor 209.165.201.6 advertisement-interval 5   ! will get into the BGP-4 table.
  neighbor 209.165.201.6 route-map IN in
  neighbor 209.165.201.6 route-map OUT out
```

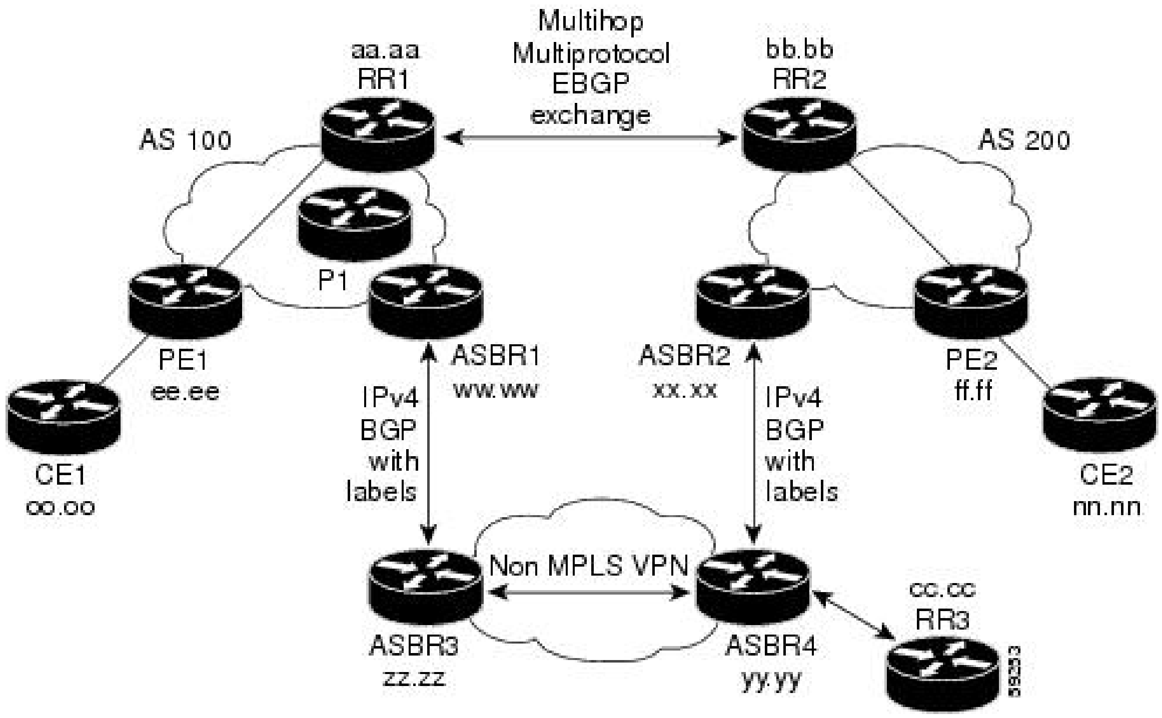
```
  neighbor 209.165.201.6 send-label
  neighbor 209.165.201.7 activate
  neighbor 209.165.201.7 advertisement-interval 5
  neighbor 209.165.201.7 route-map IN in
  neighbor 209.165.201.7 route-map OUT out
  neighbor 209.165.201.7 send-label
  no auto-summary
  no synchronization
  exit-address-family
 !
 address-family vpnv4
  neighbor 192.0.2.1 activate
  neighbor 192.0.2.1 send-community extended
  exit-address-family
  !
ip default-gateway 3.3.0.1
 ip classless
 !
 access-list 1 permit 209.165.202.129 log          !Setting up the access lists
 access-list 2 permit 203.0.113.1 log
 access-list 3 permit 192.0.2.1 log
 access-list 4 permit 10.0.0.1 log

 route-map IN permit 11                      !Setting up the route maps
  match ip address 2
  match mpls-label
 !
 route-map IN permit 12
  match ip address 4
 !
 route-map OUT permit 10
  match ip address 1
  set mpls-label
 !
 route-map OUT permit 13
  match ip address 3
 end
```

Configuration Examples: Inter-AS Using BGP to Distribute Routes and MPLS Labels Over a Non MPLS VPN Service Provider

The figure shows two MPLS VPN service providers that are connected through a non MPLS VPN service provider. The autonomous system in the middle of the network is configured as a backbone autonomous system that uses Label Distribution Protocol (LDP) or Tag Distribution Protocol (TDP) to distribute MPLS labels. You can also use traffic engineering tunnels instead of TDP or LDP to build the LSP across the non MPLS VPN service provider.

Figure 22: Distributing Routes and MPLS Labels Over a Non MPLS VPN Service Provider



Configuration examples for Inter-AS using BGP to distribute routes and MPLS labels over a non MPLS VPN service provider included in this section are as follows:

Configuration Example: Route Reflector 1 (Non MPLS VPN Service Provider)

The configuration example for RR1 specifies the following:

- RR1 exchanges VPNv4 routes with RR2 using multiprotocol, multihop EBGP.
- The VPNv4 next hop information and the VPN label are preserved across the autonomous systems.
- RR1 reflects to PE1:
  - The VPNv4 routes learned from RR2
  - The IPv4 routes and MPLS labels learned from ASBR1

```
ip subnet-zero
 ip cef
 !
 interface Loopback0
  ip address 10.0.0.1 255.255.255.255
  no ip directed-broadcast
 !
 interface Serial1/2
  ip address 209.165.201.8 255.0.0.0
  no ip directed-broadcast
  clockrate 124061
 !
 router ospf 10
  log-adjacency-changes
```

```
  auto-cost reference-bandwidth 1000
  network 10.0.0.1 0.0.0.0 area 100
  network 209.165.201.9 0.255.255.255 area 100
 !
router bgp 100
  bgp cluster-id 1
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 203.0.113.1 remote-as 100
  neighbor 203.0.113.1 update-source Loopback0
  neighbor 209.165.200.225 remote-as 100
  neighbor 209.165.200.225 update-source Loopback0
  neighbor 192.0.2.1 remote-as 200
  neighbor 192.0.2.1 ebgp-multihop 255
  neighbor 192.0.2.1 update-source Loopback0
  no auto-summary
  !
 address-family ipv4
  neighbor 203.0.113.1 activate
  neighbor 203.0.113.1 route-reflector-client                !IPv4+labels session to PE1
  neighbor 203.0.113.1 send-label
  neighbor 209.165.200.225 activate
  neighbor 209.165.200.225 route-reflector-client                !IPv4+labels session to
ASBR1
  neighbor 209.165.200.225 send-label
  no neighbor 192.0.2.1 activate
  no auto-summary
  no synchronization
  exit-address-family
  !
address-family vpnv4
  neighbor 203.0.113.1 activate
  neighbor 203.0.113.1 route-reflector-client               !VPNv4 session with PE1
  neighbor 203.0.113.1 send-community extended
  neighbor 192.0.2.1 activate
  neighbor 192.0.2.1 next-hop-unchanged                    !MH-VPNv4 session with RR2
  neighbor 192.0.2.1 send-community extended                 with next-hop-unchanged
  exit-address-family
 !
 ip default-gateway 3.3.0.1
 no ip classless
 !
 snmp-server engineID local 00000009020000D0584B25C0
 snmp-server community public RO
 snmp-server community write RW
 no snmp-server ifindex persist
 snmp-server packetsize 2048
 !
 end
```

Configuration Example: ASBR1 (Non MPLS VPN Service Provider)

ASBR1 exchanges IPv4 routes and MPLS labels with ASBR2.

In this example, ASBR1 uses route maps to filter routes.

- A route map called OUT specifies that ASBR1 should distribute the PE1 route (ee.ee) with labels and the RR1 route (aa.aa) without labels.
- A route map called IN specifies that ASBR1 should accept the PE2 route (ff.ff) with labels and the RR2 route (bb.bb) without labels.

```
ip subnet-zero
 ip cef distributed
 mpls label protocol tdp
 !
 interface Loopback0
  ip address  209.165.200.225 255.255.255.255
  no ip directed-broadcast
  no ip route-cache
  no ip mroute-cache
 !
 interface Serial3/0/0
  ip address 209.165.201.7 255.0.0.0
  no ip directed-broadcast
  ip route-cache distributed
 !
 interface Ethernet0/3
  ip address 209.165.201.18 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
  mpls label protocol ldp
  mpls ip
 !
router ospf 10
  log-adjacency-changes
  auto-cost reference-bandwidth 1000
  redistribute connected subnets
  passive-interface Serial3/0/0
  network  209.165.200.225 0.0.0.0 area 100
  network dd.0.0.0 0.255.255.255 area 100

 router bgp 100
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 10.0.0.1 remote-as 100
  neighbor 10.0.0.1 update-source Loopback0
  neighbor kk.0.0.1 remote-as 200
  no auto-summary
 !
  address-family ipv4
  redistribute ospf 10                      ! Redistributing IGP into BGP
  neighbor 10.0.0.1 activate             ! so that PE1 & RR1 loopbacks
  neighbor 10.0.0.1 send-label           ! get into BGP table
  neighbor 209.165.201.3 activate
  neighbor 209.165.201.3 advertisement-interval 5
  neighbor 209.165.201.3 send-label
  neighbor 209.165.201.3 route-map IN in    ! Accepting routes specified in route map IN
  neighbor 209.165.201.3 route-map OUT out  ! Distributing routes specified in route map
OUT
  no auto-summary
  no synchronization
  exit-address-family
 !
 ip default-gateway 3.3.0.1
 ip classless
 !
 access-list 1 permit 203.0.113.1 log
 access-list 2 permit 209.165.202.129 log
 access-list 3 permit 10.0.0.1 log
 access-list 4 permit 192.0.2.1 log
 !
 route-map IN permit 10
  match ip address 2
  match mpls-label
 !
```

```
 route-map IN permit 11
  match ip address 4
 !
 route-map OUT permit 12
  match ip address 3
 !
 route-map OUT permit 13
  match ip address 1
  set mpls-label
 !
 end
```

Configuration Example: Route Reflector 2 (Non MPLS VPN Service Provider)

RR2 exchanges VPNv4 routes with RR1 using multihop, multiprotocol EBGP. This configuration also specifies that the next hop information and the VPN label are preserved across the autonomous systems.

```
ip subnet-zero
 ip cef
 !
 interface Loopback0
  ip address 192.0.2.1 255.255.255.255
  no ip directed-broadcast
 !
 interface Serial1/1
  ip address 209.165.201.10 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
 !
 router ospf 20
  log-adjacency-changes
  network 192.0.2.1 0.0.0.0 area 200
  network 209.165.201.20 0.255.255.255 area 200
 !
router bgp 200
  bgp cluster-id 1
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 10.0.0.1 remote-as 100
  neighbor 10.0.0.1 ebgp-multihop 255
  neighbor 10.0.0.1 update-source Loopback0
  neighbor 209.165.202.129 remote-as 200
  neighbor 209.165.202.129 update-source Loopback0
  no auto-summary
  !
  address-family vpnv4
  neighbor  10.0.0.1 activate
  neighbor  10.0.0.1 next-hop-unchanged              !MH vpnv4 session with RR1
  neighbor  10.0.0.1 send-community extended              !with next-hop-unchanged
  neighbor  209.165.202.129 activate
  neighbor  209.165.202.129 route-reflector-client          !vpnv4 session with PE2
  neighbor  209.165.202.129 send-community extended
  exit-address-family
 !
 ip default-gateway 3.3.0.1
 no ip classless
 !
 end
```

Configuration Examples: ASBR2 (Non MPLS VPN Service Provider)

ASBR2 exchanges IPv4 routes and MPLS labels with ASBR1. However, in contrast to ASBR1, ASBR2 does not use the RR to reflect IPv4 routes and MPLS labels to PE2. ASBR2 redistributes the IPv4 routes and MPLS labels learned from ASBR1 into IGP. PE2 can now reach these prefixes.

```
ip subnet-zero
 ip cef
 !
 mpls label protocol tdp
 !
 interface Loopback0
  ip address 209.165.200.226 255.255.255.255
  no ip directed-broadcast
 !
 interface Ethernet0/1
  ip address 209.165.201.11 255.0.0.0
  no ip directed-broadcast
 !
 interface Ethernet1/2
  ip address 209.165.201.4 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
  mpls label protocol tdp
  mpls ip
  !
 router ospf 20
  log-adjacency-changes
  auto-cost reference-bandwidth 1000
  redistribute connected subnets
  redistribute bgp 200 subnets            !redistributing the routes learned from
  passive-interface Ethernet0/1                !ASBR2 (EBGP+labels session) into IGP
  network 209.165.200.226 0.0.0.0 area 200         !so that PE2 will learn them
  network 209.165.201.5 0.255.255.255 area 200
  !
 router bgp 200
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 192.0.2.1 remote-as 200
  neighbor 192.0.2.1 update-source Loopback0
  neighbor 209.165.201.21 remote-as 100
  no auto-summary
 !
 address-family ipv4                          ! Redistributing IGP into BGP

redistribute ospf 20                         ! so that PE2 & RR2 loopbacks
  neighbor 209.165.201.21 activate                 !  will get into the BGP-4 table
  neighbor 209.165.201.21 advertisement-interval 5
  neighbor 209.165.201.21 route-map IN in
  neighbor 209.165.201.21 route-map OUT out
  neighbor 209.165.201.21 send-label
  no auto-summary
  no synchronization
  exit-address-family
  !
 address-family vpnv4
  neighbor 192.0.2.1 activate
  neighbor 192.0.2.1 send-community extended
  exit-address-family
  !
 ip default-gateway 3.3.0.1
 ip classless
 !
 access-list 1 permit 209.165.202.129 log
```

```
 access-list 2 permit 203.0.113.1 log
 access-list 3 permit 192.0.2.1 log
 access-list 4 permit 10.0.0.1 log
 !
 route-map IN permit 11
  match ip address 2
  match mpls-label
 !
 route-map IN permit 12
  match ip address 4
 !
 route-map OUT permit 10
  match ip address 1
  set mpls-label
 !
 route-map OUT permit 13
  match ip address 3
 !
 end
```

Configuration Example: ASBR3 (Non MPLS VPN Service Provider)

ASBR3 belongs to a non MPLS VPN service provider. ASBR3 exchanges IPv4 routes and MPLS labels with ASBR1. ASBR3 also passes the routes learned from ASBR1 to ASBR3 through RR3.

Note Do not redistribute EBGP routes learned into IBG if you are using IBGP to distribute the routes and labels. This is not a supported configuration.

```
ip subnet-zero
 ip cef
 !
 interface Loopback0
  ip address 209.165.200.227 255.255.255.255
  no ip directed-broadcast
  no ip route-cache
  no ip mroute-cache
 !
 ip routing
mpls label protocol ldp
mpls ldp router-id Loopback0 force

interface GigabitEthernet1/0/1
ip address 209.165.201.12 255.0.0.0

interface TenGigabitEthernet1/1/1
no switchport
ip address 209.165.201.3 255.0.0.0
load-interval 30
mpls ip

 !
 router ospf 30
 log-adjacency-changes
 auto-cost reference-bandwidth 1000
 redistribute connected subnets
 network 209.165.200.227 0.0.0.0 area 300
 network 209.165.201.13 0.255.255.255 area 300
 !
 router bgp 300
  bgp log-neighbor-changes
```

```
 timers bgp 10 30
 neighbor 10.0.0.3 remote-as 300
 neighbor 10.0.0.3 update-source Loopback0
 neighbor 209.165.201.7 remote-as 100
 no auto-summary
 !
 address-family ipv4
 neighbor 10.0.0.3activate            ! IBGP+labels session with RR3
 neighbor 10.0.0.3 send-label
 neighbor 209.165.201.7 activate                ! EBGP+labels session with ASBR1
 neighbor 209.165.201.7 advertisement-interval 5
 neighbor 209.165.201.7 send-label
neighbor 209.165.201.7 route-map IN in
neighbor 209.165.201.7 route-map OUT out
no auto-summary
 no synchronization
 exit-address-family
!
ip classless
!
access-list 1 permit 203.0.113.1 log
access-list 2 permit 209.165.202.129 log
access-list 3 permit 10.0.0.1 log
access-list 4 permit 192.0.2.1 log
!
route-map IN permit 10
 match ip address 1
  match mpls-label
!
route-map IN permit 11
   match ip address 3
!
route-map OUT permit 12
 match ip address 2
  set mpls-label
!
route-map OUT permit 13
   match ip address 4
!
ip default-gateway 3.3.0.1
ip classless
!
end
```

Configuration Example: Route Reflector 3 (Non MPLS VPN Service Provider)

RR3 is a non MPLS VPN RR that reflects IPv4 routes with MPLS labels to ASBR3 and ASBR4.

```
ip subnet-zero
 mpls label protocol tdp
 mpls traffic-eng auto-bw timers
 no mpls ip
 !
 interface Loopback0
  ip address 10.0.0.3 255.255.255.255
  no ip directed-broadcast
 !
 interface POS0/2
  ip address 209.165.201.15 255.0.0.0
  no ip directed-broadcast
  no ip route-cache cef
  no ip route-cache
  no ip mroute-cache
  crc 16
```

```
 clock source internal
!
router ospf 30
 log-adjacency-changes
 network 10.0.0.3 0.0.0.0 area 300
 network 209.165.201.16 0.255.255.255 area 300
!
router bgp 300
 bgp log-neighbor-changes
 neighbor 209.165.201.2 remote-as 300
 neighbor 209.165.201.2 update-source Loopback0
 neighbor 209.165.200.227 remote-as 300
 neighbor 209.165.200.227 update-source Loopback0
 no auto-summary
 !
address-family ipv4
 neighbor 209.165.201.2 activate
 neighbor 209.165.201.2 route-reflector-client
 neighbor 209.165.201.2 send-label               ! IBGP+labels session with ASBR3
 neighbor 209.165.200.227 activate
 neighbor 209.165.200.227 route-reflector-client
 neighbor 209.165.200.227 send-label               ! IBGP+labels session with ASBR4
 no auto-summary
 no synchronization
 exit-address-family
!
ip default-gateway 3.3.0.1
ip classless
!
end
```

Configuration Example: ASBR4 (Non MPLS VPN Service Provider)

ASBR4 belongs to a non MPLS VPN service provider. ASBR4 and ASBR3 exchange IPv4 routes and MPLS labels by means of RR3.

Note Do not redistribute EBGP routes learned into IBG if you are using IBGP to distribute the routes and labels. This is not a supported configuration.

```
ip subnet-zero
 ip cef distributed
 !
 interface Loopback0
  ip address 209.165.201.2 255.255.255.255
  no ip directed-broadcast
  no ip route-cache
  no ip mroute-cache
 !
 interface Ethernet0/2
  ip address 209.165.201.21 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
 !
 ip routing
mpls label protocol ldp
mpls ldp router-id Loopback0 force


interface GigabitEthernet1/0/1
ip address 209.165.201.17 255.0.0.0
```

```
interface TenGigabitEthernet1/1/1
no switchport
ip address 209.165.201.14 255.0.0.0
load-interval 30
mpls ip

 !
 router ospf 30
  log-adjacency-changes
  auto-cost reference-bandwidth 1000
  redistribute connected subnets
 passive-interface Ethernet0/2
  network 209.165.201.2 0.0.0.0 area 300
  network 209.165.201.16 0.255.255.255 area 300
  network 209.165.201.13 0.255.255.255 area 300
  !
 router bgp 300
  bgp log-neighbor-changes
  timers bgp 10 30
  neighbor 10.0.0.3 remote-as 300
  neighbor 10.0.0.3 update-source Loopback0
  neighbor 209.165.201.11 remote-as 200
  no auto-summary
  !
  address-family ipv4
  neighbor 10.0.0.3 activate
  neighbor 10.0.0.3 send-label
  neighbor 209.165.201.11 activate
  neighbor 209.165.201.11 advertisement-interval 5
  neighbor 209.165.201.11 send-label
  neighbor 209.165.201.11 route-map IN in
  neighbor 209.165.201.11 route-map OUT out
 no auto-summary
  no synchronization
  exit-address-family
 !
 ip classless
 !
 access-list 1 permit 209.165.202.129 log
 access-list 2 permit 203.0.113.1 log
 access-list 3 permit 192.0.2.1 log
 access-list 4 permit 10.0.0.1 log
 !
 route-map IN permit 10
  match ip address 1
   match mpls-label
 !
 route-map IN permit 11
    match ip address 3
 !
 route-map OUT permit 12
  match ip address 2
   set mpls-label
 !
 route-map OUT permit 13
    match ip address 4
 !
 ip default-gateway 3.3.0.1
 ip classless
 !
 end
```

Feature History for Configuring MPLS VPN Inter-AS IPv4 BGP Label Distribution

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	MPLS VPN Inter-AS IPv4 BGP Label Distribution	This feature enables you to set up a Virtual Private Network (VPN) service provider network. In this network, the Autonomous System Boundary Routers (ASBRs) exchange IPv4 routes with Multiprotocol Label Switching (MPLS) labels of the provider edge (PE) routers.

Use Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.