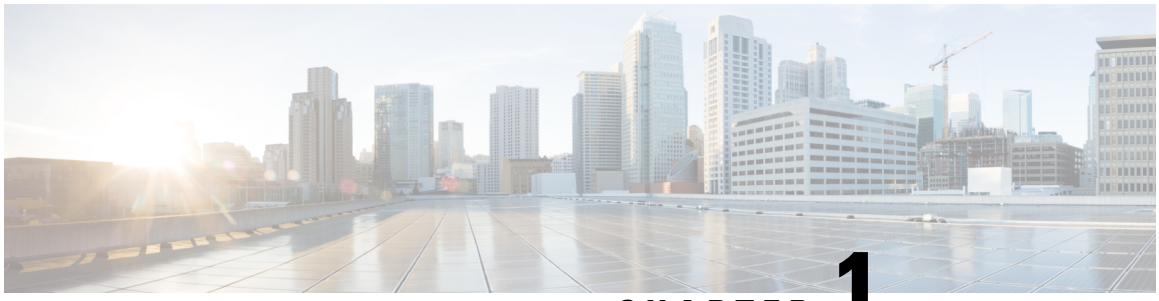
Multiprotocol Label Switching Configuration Guide, Cisco IOS XE Dublin 17.12.x (Catalyst 9400 Switches)

First Published: 2023-07-31

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 (MPLS) and how to configure it on Cisco switches.

Restrictions for Multiprotocol Label Switching

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

Information about 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.

Functions of Label Switching

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.

After 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 MPLS 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, 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 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 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.

LAN MACsec over MPLS

From the Cisco IOS XE Dublin 17.11.1 release, MPLS packets can be encrypted with a MACsec tag. Media Access Control security (MACsec) protocol is a IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. To use MPLS with MACsec both MPLS and MACsec need to be configured on both the devices. When an MPLS packet is forwarded by one device, the MPLS packet is treated as the inner payload and is encrypted with a MACsec tag. This encrypted packet is then securely forwarded to the other device. MACsec encryption safeguards the network against a range of attacks including denial of service, intrusion, man-in-the-middle and eavesdropping. The other device receives the MACsec tagged MPLS packet. It decrypts the MACsec tag and forwards the MPLS packet.

Note LAN MACsec over MPLS is not supported on Silicon One devices. It is not supported on Catalyst 9500X Series Switches and Catalyst 9600 Supervisor Series 2 Module.

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

Before you configure MPLS switching on Cisco switches, ensure that the Cisco Express Forwarding (CEF) is enabled.

Note The command **ip unnumbered** 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:**	Specifies the label distribution protocol for the platform.

	Command or Action	Purpose
	`Device(config)# mpls label protocol ldp`	

Configuring a Switch for MPLS Forwarding

Before you configure MPLS forwarding on Cisco switches, ensure that the forwarding of IPv4 packets is enabled.

Note The command **ip unnumbered** 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.

How to Verify Multiprotocol Label Switching Configuration

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

Verifying Configuration of MPLS Switching

To verify whether Cisco Express Forwarding has been configured properly, enter 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 whether MPLS forwarding has been configured properly, enter 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.	

Feature History for Multiprotocol Label Switching

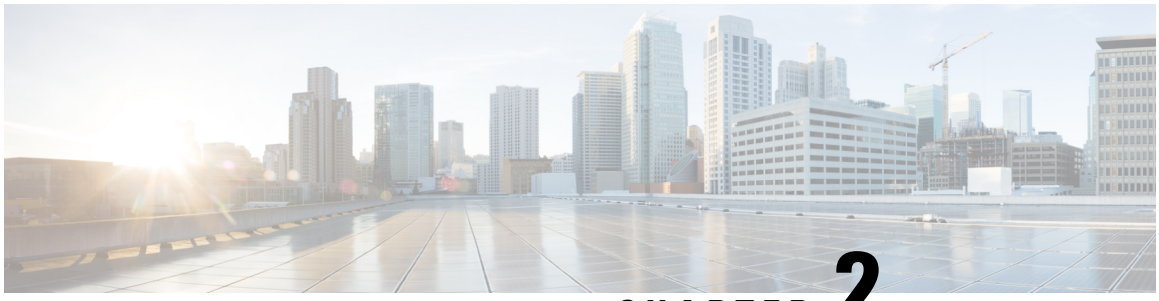
This table provides release and related information for features explained in this module.

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 16.6.1	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.
Cisco IOS XE Cupertino 17.7.1	Multiprotocol Label Switching	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

Release	Feature	Feature Information
Cisco IOS XE Dublin 17.11.1	LAN MACsec over MPLS	LAN MACsec over MPLS allows MPLS packets to be encrypted with a MACsec tag. This allows for the flexibility and capability of MPLS to be used with the encryption and security of MACsec.

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 MPLS Layer 3 VPN

This document describes how to configure Multiprotocol Label Switching (MPLS) Virtual Private Networks (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.

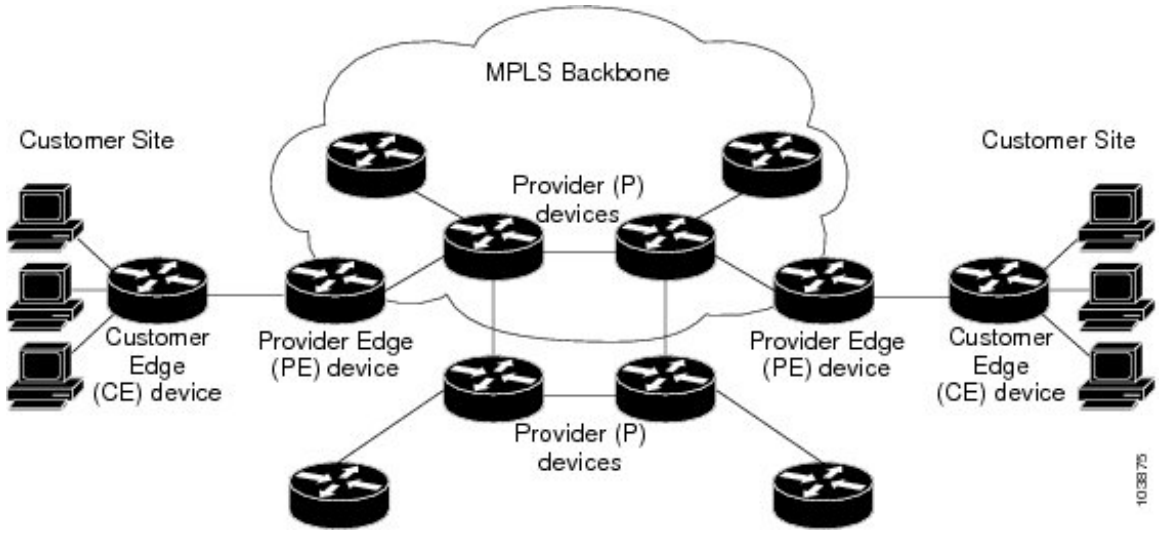
MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the service provider's edge device that provides services to the customer site needs to be updated.

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 1: 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: • 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**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
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.

	Command or Action	Purpose
		• The *route-target-ext-community* argument adds the route-target extended community attributes to the VRF's list of import, 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

PE Configuration	CE Configuration
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	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

Example: Configuring an MPLS Virtual Private Network Using Static Routes

PE Configuration	CE Configuration
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	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

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	

PE Configuration	CE Configuration

<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>	

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 9400 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

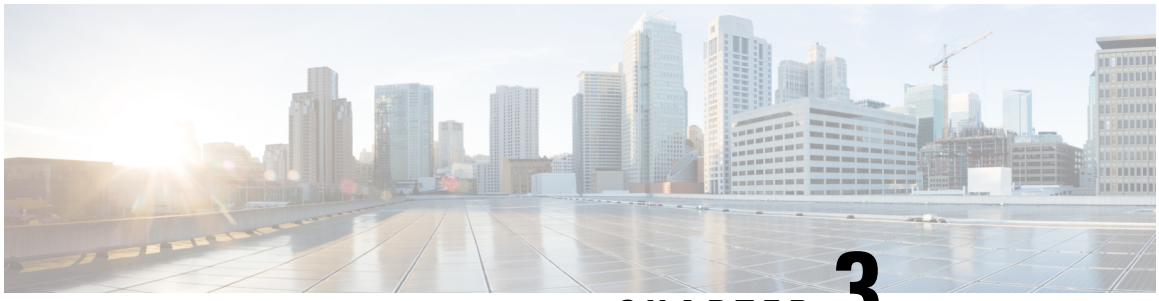
This table provides release and related information for features explained in this module.

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 16.6.1	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.
Cisco IOS XE Gibraltar 16.11.1	BGP PE-CE support for MPLS Layer 3 VPNs	Support for BGP as a routing protocol between the provider edge (PE) device and the customer edge (CE) device was introduced.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	MPLS Virtual Private Networks	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 3

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.
- If pairing of equal cost routing is more than 64 unique paths, the routes are not learnt and traffic is dropped.

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 valid values for the **maximum-paths** command range from 1 to 32. However, the maximum value that can be configured is 2.

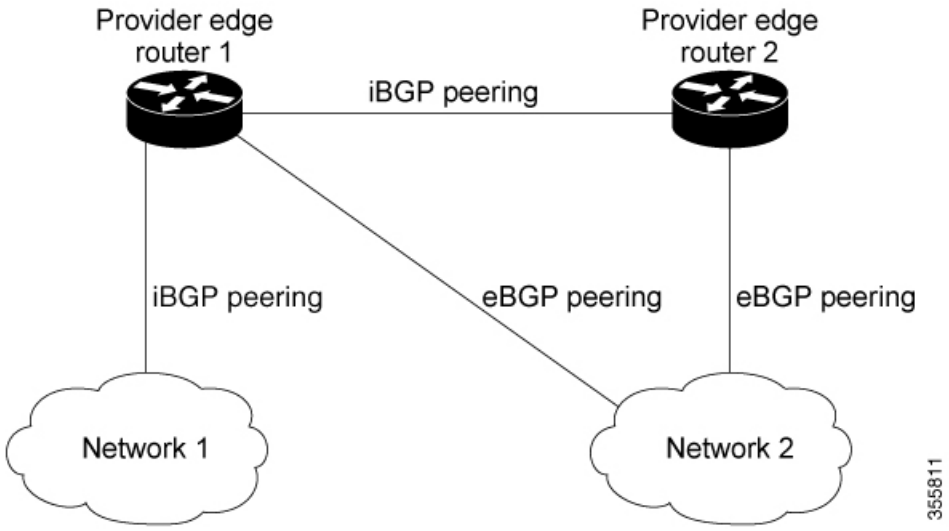
Load balancing over the multipaths is performed by CEF. CEF load balancing is configured on a per-packet round robin or on a per session (source and destination pair) basis. For information about CEF, see IP Switching Cisco Express Forwarding Configuration Guide. The BGP Multipath Load Sharing for Both 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 2: Service Provider BGP MPLS Network Connected to PE Routers



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 in an MPLS VPN

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:**	Specifies the link-local address over which the peering is to occur.

	Command or Action	Purpose
	`Device(config-router)# neighbor FE80::1234:BFF:FE0E:A471 update-source Gigabitethernet 1/0/0`	
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.
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 and 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

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

Example: Configuring eBGP and iBGP Multipath Load Sharing

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

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

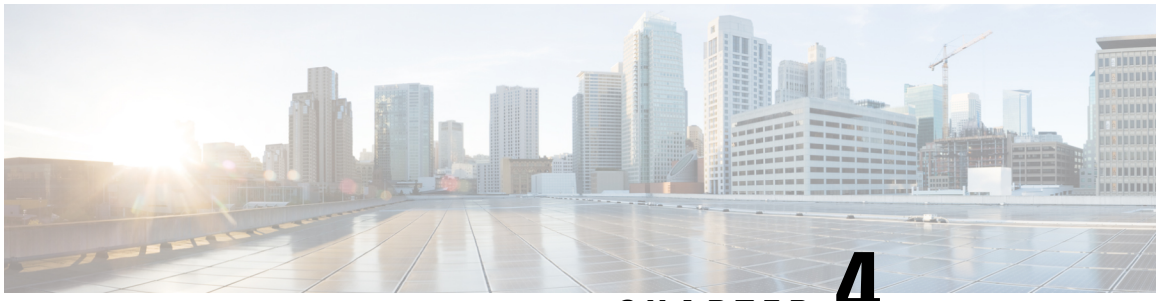
These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	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.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	BGP Multipath Load Sharing for Both eBGP and iBGP in an MPLS-VPN	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 4

Configuring EIGRP MPLS VPN PE-CE

Prerequisites for MPLS VPN Support for EIGRP Between PE and CE

- Configure MPLS Layer 3 VPNs.
- Configure the Border Gateway Protocol (BGP) in the network core.

Information About MPLS VPN Support for EIGRP Between PE and CE

How to Configure MPLS VPN Support for EIGRP Between PE and CE

This section provides information about how to configure MPLS VPN support for EIGRP bbetween PE and CE:

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

To configure PE-to-CE routing sessions that use EIGRP, perform this task.

Before you begin

Configure the PE device with the same routing protocol that the CE device uses.

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 10`	Enters router configuration mode, and creates a BGP routing process.
Step 4	**no synchronization** **Example:** `Device(config-router)# no synchronization`	Configures BGP to send advertisements without waiting to synchronize with the IGP.
Step 5	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 10.0.0.1 remote-as 10`	Establishes peering with the specified neighbor or peer group. • In this step, you are establishing an iBGP session with the PE device that is connected to the CE device at the other CE site.
Step 6	**neighbor** *ip-address* **update-source loopback** *interface-number* **Example:** `Device(config-router)# neighbor 10.0.0.1 update-source loopback 0`	Configures BGP to use any operational interface for TCP connections. • This configuration step is not required. However, the BGP routing process will be less susceptible to the effects of interface or link flapping.
Step 7	**address-family vpnv4** **Example:** `Device(config-router)# address-family vpnv4`	Enters address family configuration mode for configuring routing sessions that use standard IPv4 address prefixes, such as BGP, RIP, and static routing sessions.

	Command or Action	Purpose
Step 8	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.0.0.1 activate`	Establishes peering with the specified neighbor or peer group. • In this step, you are activating the exchange of VPNv4 routing information between the PE devices.
Step 9	**neighbor** *ip-address* **send-community extended** **Example:** `Device(config-router-af)# neighbor 10.0.0.1 send-community extended`	Configures the local device to send extended community attribute information to the specified neighbor. • This step is required for the exchange of EIGRP extended community attributes.
Step 10	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and enters router configuration mode.
Step 11	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv4 vrf RED`	Configures an IPv4 address family for the EIGRP VRF and enters address family configuration mode. • An address-family VRF needs to be configured for each EIGRP VRF that runs between the PE and CE devices.
Step 12	**redistribute eigrp** *as-number* [**metric** *metric-value*] [**route-map** *map-name*] **Example:** `Device(config-router-af)# redistribute eigrp 101`	Redistributes the EIGRP VRF into BGP. • The autonomous system number from the CE network is configured in this step.
Step 13	**no synchronization** **Example:** `Device(config-router-af)# no synchronization`	Configures BGP to send advertisements without waiting to synchronize with the IGP.
Step 14	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and enters router configuration mode.
Step 15	**end** **Example:**	Exits router configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-router)# end`	

Configuring EIGRP Redistribution in the MPLS VPN

Perform this task on every PE device that provides VPN services to enable EIGRP redistribution in the MPLS VPN.

Before you begin

The metric must be configured for routes from external EIGRP autonomous systems and non-EIGRP networks before these routes can be redistributed into an EIGRP CE device. The metric can be configured in the redistribute statement using the **redistribute** (IP) command or can be configured with the **default-metric** (EIGRP) command. If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route will not be advertised to the CE device.

Note Redistribution between native EIGRP VRFs is not supported. This is designed behavior.

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 eigrp** *as-number* **Example:** `Device(config)# router eigrp 1`	Enters router configuration mode and creates an EIGRP routing process. • The EIGRP routing process for the PE device is created in this step.
Step 4	**address-family ipv4** [**multicast** | **unicast** | **vrf** *vrf-name*] **Example:** `Device(config-router)# address-family ipv4 vrf RED`	Enters address-family configuration mode and creates a VRF. • The VRF name must match the VRF name that was created in the previous section.
Step 5	**network** *ip-address wildcard-mask*	Specifies the network for the VRF.

	Command or Action	Purpose
	Example: `Device(config-router-af)# network 172.16.0.0 0.0.255.255`	• The network statement is used to identify which interfaces to include in EIGRP. The VRF must be configured with addresses that fall within the wildcard-mask range of the network statement.
Step 6	**redistribute bgp** {*as-number*} [**metric** *bandwidth delay reliability load mtu*] [**route-map** *map-name*] **Example:** `Device(config-router-af)# redistribute bgp 10 metric 10000 100 255 1 1500`	Redistributes BGP into the EIGRP. • The autonomous system number and metric of the BGP network are configured in this step. BGP must be redistributed into EIGRP for the CE site to accept the BGP routes that carry the EIGRP information. A metric must also be specified for the BGP network and is configured in this step.
Step 7	**autonomous-system** *as-number* **Example:** `Device(config-router-af)# autonomous-system 101`	Specifies the autonomous system number of the EIGRP network for the customer site.
Step 8	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits address family configuration mode and enters router configuration mode.
Step 9	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

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 VPN Support for EIGRP Between PE and CE

This section provides the configuration examples for MPLS VPN support for EIGRP between PE and CE:

Example: Configuring an MPLS VPN Using EIGRP

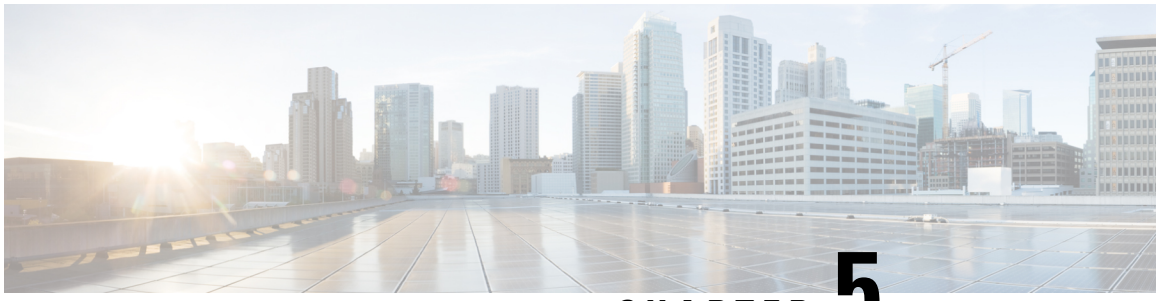
PE Configuration	CE Configuration
ip vrf vpn1 rd 100:1 route-target export 100:1 route-target import 100:1 ! ip cef mpls ldp router-id Loopback0 force mpls label protocol ldp ! interface Loopback0 ip address 10.0.0.1 255.255.255.255 interface FastEthernet0/0/0 ip vrf forwarding vpn1 ip address 34.0.0.2 255.0.0.0 no cdp enable interface FastEthernet1/1/0 ip address 30.0.0.1 255.0.0.0 mpls label protocol ldp mpls ip router eigrp 1000 auto-summary ! address-family ipv4 vrf vpn1 redistribute bgp 100 metric 10000 100 255 1 1500 network 34.0.0.0 distribute-list 20 in no auto-summary autonomous-system 1000 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 eigrp no auto-summary no synchronization exit-address-family	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 FastEthernet0/0/0 ip address 34.0.0.1 255.0.0.0 no cdp enable ! router eigrp 1000 network 34.0.0.0 auto-summary

Feature History for MPLS VPN Support for EIGRP Between PE and CE

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 MPLS VPN Support for EIGRP Between PE and CE

Release	Feature Name	Feature Information
Cisco IOS XE Fuji 16.9.1	MPLS VPN Support for EIGRP Between PE and CE	The MPLS VPN Support for EIGRP Between PE and CE feature allows service providers to configure the Enhanced Interior Gateway Routing Protocol (EIGRP) between provider edge (PE) and customer edge (CE) devices in a Multiprotocol Label Switching (MPLS) virtual private network (VPN) and offer MPLS VPN services to those customers that require native support for EIGRP.
Cisco IOS XE Cupertino 17.7.1	MPLS VPN Support for EIGRP Between PE and CE	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.



CHAPTER 5

Configuring Ethernet-over-MPLS (EoMPLS)

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 Ethernet-over-MPLS

The following sections list the restrictions for EoMPLS port mode and EoMPLS VLAN mode.

Restrictions for Ethernet-over-MPLS Port Mode

- Ethernet Flow Point is not supported.

- Quality of Service (QoS): Customer differentiated services code point (DSCP) re-marking is not supported with virtual private wire service (VPWS) and EoMPLS.
- Virtual Circuit Connectivity Verification (VCCV) ping with explicit null is not supported.
- Layer 2 Protocol Tunneling CLI is not supported.
- Flow-Aware Transport (FAT) Pseudowire Redundancy is supported only in Protocol-CLI mode. Supported load-balancing parameters are Source IP, Source MAC address, Destination IP, and Destination MAC address.
- MPLS QoS is supported only in pipe and uniform mode. Default mode is pipe mode.
- Both 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 available between CE devices.
- By default, EoMPLS PW tunnels all the protocols such as Cisco Discovery Protocol and Spanning Tree Protocol (STP). EoMPLS PW cannot perform selective protocol tunneling as part of L2 Protocol Tunneling CLI.

Restrictions for EoMPLS VLAN Mode

- Virtual circuit will not work if the same interworking type is not configured on PE devices.
- Untagged traffic is not supported as incoming traffic.
- Xconnect mode cannot be enabled on Layer 2 subinterfaces because multiplexer user-network interface (MUX UNI) is not supported.
- Xconnect mode cannot be configured on subinterfaces if it is enabled on the main interface for port-to-port transport.
- FAT can be configured on Protocol CLI mode only.
- In VLAN mode EoMPLS, only those packets encrypted with the dot1q in clear by the CE device will be processed by the PE device.
- QoS: Customer DSCP Remarking is not supported with VPWS and EoMPLS.
- MPLS QoS is supported in pipe and uniform mode. Default mode is pipe mode.
- In VLAN mode EoMPLS, Cisco Discovery Protocol packets from the CE will be processed by the PE, but will not be carried over the EoMPLS virtual circuit, whereas in port mode, Cisco Discovery Protocol packets from the CE will be carried over the virtual circuit.
- Only Ethernet and VLAN interworking types are supported.
- L2 Protocol Tunneling is not supported.

Information About Ethernet-over-MPLS

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.

The following modes are 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.
- VLAN mode: Transports Ethernet traffic from a source 802.1Q VLAN to a destination 802.1Q VLAN through a single virtual circuit over an MPLS network. VLAN mode uses virtual circuit type 5 as the default (does not transport dot1q tag); however, uses virtual circuit type 4 (transports dot1 tag) if the remote PE does not support virtual circuit type 5 for subinterface-based (VLAN-based) EoMPLS.

Interworking between EoMPLS port mode and EoMPLS VLAN mode: If EoMPLS port mode is configured on a local PE and EoMPLS VLAN mode on a remote PE, then the customer edge (CE) Layer 2 switchport interface must be configured as an *access* on the port mode side and the Spanning Tree Protocol must be disabled on the VLAN mode side of the CE device.

The maximum transmission unit (MTU) of all the intermediate links between PEs must be able to carry the largest Layer 2 packet received on ingress PE.

Starting with the Cisco IOS XE Bengaluru 17.6.1 release, you can forward Link Aggregation Control Protocol (LACP) and Port Aggregation Protocol (PAgP) packets over Ethernet-over-MPLS Pseudowire in the Port mode.

How to Configure Ethernet-over-MPLS

EoMPLS can be configured in the port mode or VLAN mode.

Configuring Ethernet-over-MPLS Port Mode

EoMPLS port mode can be configured using either the Xconnect mode or 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.

	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 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.
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.
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:**	Ensures that the device does not send keepalive messages.

	Command or Action	Purpose
	`Device(config-if)# no keepalive`	
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.
Step 13	**member** *interface-id* **Example:** `Device(config-if-xconn)# member TenGigabitEthernet1/0/21`	Specifies interface that forms an L2VPN cross connect.

	Command or Action	Purpose
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.

Configuring Ethernet-over-MPLS VLAN Mode

EoMPLS VLAN mode can be configured using either the Xconnect mode or protocol-CLI method.

Xconnect Mode

To configure EoMPLS VLAN 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.

	Command or Action	Purpose
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	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**interface** *interface-id.subinterface* **Example:** `Device(config)# interface TenGigabitEthernet1/0/36.1105`	Defines the subinterface to be configured, and enters subinterface configuration mode.
Step 9	**encapsulation dot1Q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1Q 1105`	Enables IEEE 802.1Q encapsulation of traffic on the subinterface.
Step 10	**xconnect** *peer-ip-addr vc-id* **encapsulation mpls** **Example:** `Device(config-subif)# xconnect 10.0.0.1 1105 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 11	**end** **Example:** `Device(config-subif-xconn)# end`	Returns to privileged EXEC mode.

Protocol CLI Method

To configure EoMPLS VLAN 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.
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.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 9	**interface** *interface-id.subinterface* **Example:** `Device(config)# interface TenGigabitEthernet1/0/36.1105`	Defines the subinterface to be configured, and enters subinterface configuration mode.
Step 10	**encapsulation dot1Q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1Q 1105`	Enables IEEE 802.1Q encapsulation of traffic on the subinterface.
Step 11	**exit** **Example:** `Device(config-subif)# exit`	Exits subinterface configuration mode and returns to interface configuration mode.
Step 12	**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 13	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies the tunneling encapsulation.
Step 14	**neighbor** *peer-ip-addr vc-id* **Example:** `Device(config-if)# neighbor 10.10.0.10 17`	Specifies the peer IP address and VC ID value of a L2VPN pseudowire.

	Command or Action	Purpose
Step 15	**l2vpn xconnect context** *context-name* **Example:** `Device(config-if)# l2vpn xconnect context vpws17`	Creates a L2VPN cross connect context, and enters Xconnect context configuration mode.
Step 16	**member** *interface-id.subinterface* **Example:** `Device(config-if-xconn)# member TenGigabitEthernet1/0/36.1105`	Specifies the subinterface that forms a L2VPN cross connect.
Step 17	**member pseudowire** *number* **Example:** `Device(config-if-xconn)# member pseudowire 17`	Specifies pseudowire interface that forms a L2VPN cross connect.
Step 18	**end** **Example:** `Device(config-if-xconn)# end`	Exits Xconnect configuration mode and returns to privileged EXEC mode.

Configuration Examples for Ethernet-over-MPLS

Figure 3: EoMPLS Topology

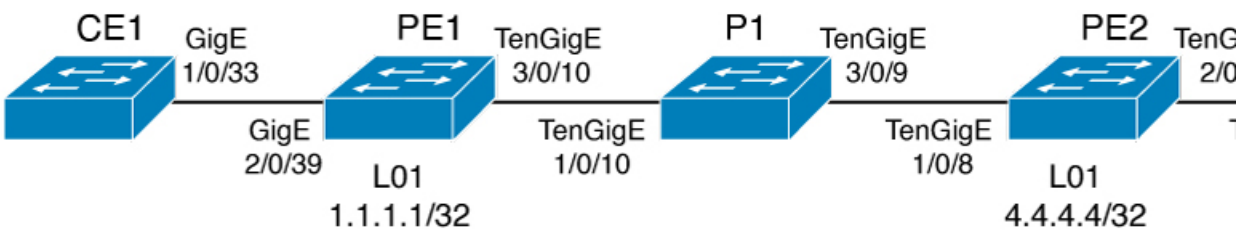


Table 2: EoMPLS Port Mode Configuration

PE Configuration	CE Configuration
mpls ip mpls label protocol ldp mpls ldp graceful-restart mpls ldp router-id loopback 1 force interface Loopback1 ip address 10.1.1.1 255.255.255.255 ip ospf 100 area 0 router ospf 100 router-id 10.1.1.1 nsf system mtu 9198 port-channel load-balance dst-ip ! interface gigabitethernet 2/0/39 no switchport no ip address no keepalive ! interface pseudowire101 encapsulation mpls neighbor 10.10.10.10 101 load-balance flow ip dst-ip load-balance flow-label both l2vpn xconnect context pw101 member pseudowire101 member gigabitethernet 2/0/39 ! interface tengigabitethernet 3/0/10 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 10.11.11.11 255.255.255.0 ip ospf 100 area 0 mpls ip mpls label protocol ldp !	interface gigabitethernet 1/0/33 switchport trunk allowed vlan 912 switchport mode trunk spanning-tree portfast trunk ! interface Vlan912 ip address 10.91.2.3 255.255.255.0 !

Table 3: EoMPLS VLAN Mode Configuration

PE Configuration	CE Configuration
interface tengigabitethernet 1/0/36 no switchport no ip address no keepalive exit ! interface tengigabitethernet 1/0/36.1105 encapsulation dot1Q 1105 exit ! interface pseudowire1105 encapsulation mpls neighbor 10.10.0.10 1105 exit ! l2vpn xconnect context vme1105 member tengigabitethernet 1/0/36.1105 member pseudowire1105 end !	interface fortygigabitethernet 1/9 switchport switchport mode trunk switchport trunk allowed vlan 1105 mtu 9216 end !

Table 4: Interworking Between EoMPLS Port Mode and EoMPLS VLAN Mode Configuration

PE Configuration: Port Mode	CE Configuration: Port Mode
interface tengigabitethernet 1/0/37 no switchport no ip address no keepalive exit ! interface pseudowire1105 encapsulation mpls neighbor 10.11.11.11 1105 exit ! l2vpn xconnect context vme1105 member tengigabitethernet 1/0/37 member pseudowire1105 end !	interface fortygigabitethernet1/10 switchport switchport mode access switchport access vlan 1105 end no spanning-tree vlan 1105 !

PE Configuration: VLAN Mode	CE Configuration: VLAN Mode
interface tengigabitethernet 1/0/36 no switchport no ip address no keepalive exit ! interface tengigabitethernet 1/0/36.1105 encapsulation dot1Q 1105 exit ! interface pseudowire1105 encapsulation mpls neighbor 10.10.0.10 1105 exit ! l2vpn xconnect context vme1105 member tengigabitethernet 1/0/36.1105 member pseudowire1105 end !	interface fortygigabitethernet 1/9 switchport switchport mode trunk switchport trunk allowed vlan 1105 mtu 9216 end no spanning-tree vlan 1105 !

Another scenario for interworking between EoMPLS port mode and EoMPLS VLAN mode is to configure the following commands on both CE devices:

- **switchport mode trunk**
- **switchport trunk allowed vlan** *vlan-id*
- **spanning-tree vlan** *vlan-id*

Data traffic will flow through by disabling STP on both CE devices, if the traffic sent is not double VLAN tagged.

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

```
Device# show mpls l2 vc vcid 1105 detail
Local interface: TenGigabitEthernet1/0/36.1105 up, line protocol up, Eth VLAN 1105 up
  Interworking type is Ethernet
  Destination address: 10.0.0.1, VC ID: 1105, VC status: up
    Output interface: Po10, imposed label stack {33 10041}
    Preferred path: not configured
    Default path: active
    Next hop: 10.10.0.1
  Create time: 00:04:09, last status change time: 00:02:13
    Last label FSM state change time: 00:02:12
  Signaling protocol: LDP, peer 10.0.0.1:0 up
    Targeted Hello: 10.0.0.10(LDP Id) -> 10.0.0.1, 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 124, remote 10041
  Group ID: local 336, remote 352
  MTU: local 9198, remote 9198
  Remote interface description:
  MAC Withdraw: sent:1, received:0
Sequencing: receive disabled, send disabled
Control Word: On (configured: autosense)
SSO Descriptor: 10.0.0.1/1105, local label: 124
Dataplane:
  SSM segment/switch IDs: 9465983/446574 (used), PWID: 109
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 following is a sample output of the **show l2vpn atom vc vcid** *vc-id* **detail** command:

```
Device# show l2vpn atom vc vcid 1105 detail
pseudowire100109 is up, VC status is up PW type: Ethernet
  Create time: 00:04:17, last status change time: 00:02:22
    Last label FSM state change time: 00:02:20
  Destination address: 10.0.0.1 VC ID: 1105
    Output interface: Po10, imposed label stack {33 10041}
    Preferred path: not configured
    Default path: active
    Next hop: 10.10.0.1
  Member of xconnect service TenGigabitEthernet1/0/36.1105-1105, group right
    Associated member TenGigabitEthernet1/0/36.1105 is up, status is up
    Interworking type is Ethernet
    Service id: 0x1f000037
  Signaling protocol: LDP, peer 10.0.0.1:0 up
    Targeted Hello: 10.0.0.10(LDP Id) -> 10.0.0.1, LDP is UP
    Graceful restart: configured and enabled
    Non stop routing: not configured and not enabled
    PWid FEC (128), VC ID: 1105
    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        124                            10041
    Group ID     336                            352
    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
SSO Descriptor: 10.0.0.1/1105, local label: 124
Dataplane:
  SSM segment/switch IDs: 9465983/446574 (used), PWID: 109
Rx Counters
  0 input transit packets, 0 bytes
  0 drops, 0 seq err
  0 MAC withdraw
Tx Counters
  0 output transit packets, 0 bytes
  0 drops
  1 MAC withdraw
```

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

```
Device# show mpls forwarding-table 10.0.0.1

Local      Outgoing   Prefix           Bytes Label   Outgoing       Next Hop
Label      Label      or Tunnel Id     Switched      interface
2049       33         10.0.0.1/32      38540         Hu2/0/30/2.1   10.0.0.2
           33         10.0.0.1/32      112236        Hu2/0/30/2.2   10.0.0.6
           33         10.0.0.1/32      46188         Hu2/0/30/2.3   10.0.0.8
```

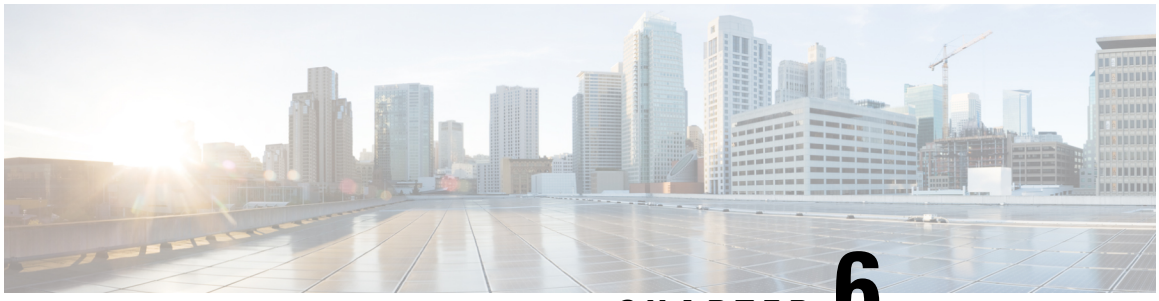
Feature History for Ethernet-over-MPLS (EoMPLS)

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	Ethernet-over-MPLS and Pseudowire Redundancy	Ethernet-over-MPLS is one of the Any Transport over MPLS (AToM) transport types. The Layer 2 VPN 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.

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 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.

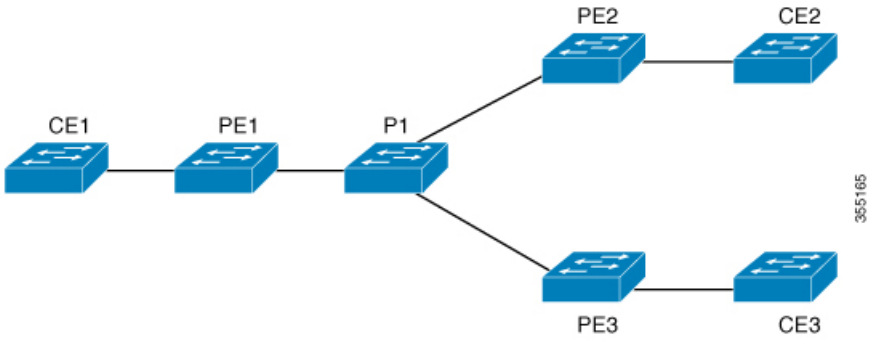
While implementing 6PE, the provider edge routers are upgraded to support 6PE, while the rest of the core network is not touched (IPv6 unaware). This implementation requires no reconfiguration of core routers because forwarding is based on labels rather than on the IP header itself. This provides a cost-effective 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 4: 6PE Topology



IPv6 Explicit Null label for 6PE

The null label is a label that is used between the penultimate Label Switch Router (LSR) and the egress LSR.

Starting with Cisco IOS XE Bengaluru 17.6.1 release, you can use the IPv6 Explicit Null Label as the VPN label to exchange IPv6 reachability information over the MPLS core. The IPv6 Explicit Null Label has a value of 2. The null label does not use the Border Gateway Protocol (BGP) to transport labels. When you use null labels the BGP labels are not exhuasted and more IPv6 prefixes can be supported.

You can configure the IPv6 Explicit Null Label by using the **label mode** [**explicit-null** | **all-explicit-null**] command in the address family configuration mode.

You can choose the **explicit-null** or the **all-explicit-null** label.

- **explicit-null** label: for directly connected IPv6 prefixes sent to BGP labelled unicast neighbors.
- **all-explicit-null** label: for all IPv6 prefixes sent to BGP labelled unciast neighbors.

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.
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:**	Adds an entry to the BGP or multiprotocol BGP neighbor table.

	Command or Action	Purpose
	`Device(config-router)# neighbor 33.33.33.33 remote-as 65001`	• *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:** `Device(config-router)# neighbor 33.33.33.33 update-source Loopback1`	Configures BGP sessions to use any operational interface for TCP connections.
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.

	Command or Action	Purpose
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.

Configuring IPV6 Explicit Null label for 6PE

To configure IPv6 explicit null label for 6PE, complete the following steps.

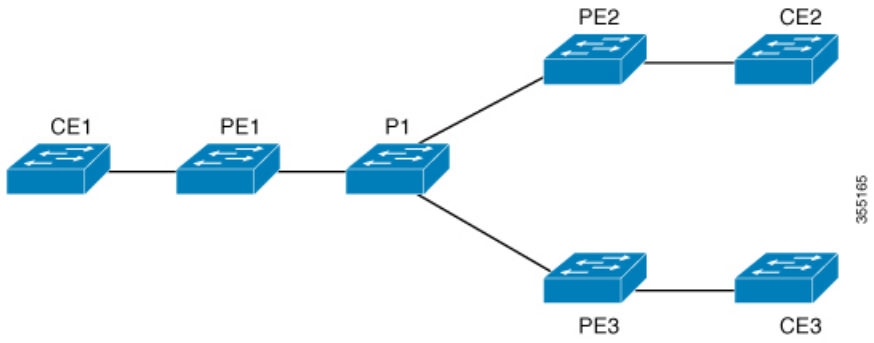
Procedure

	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.
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.

	Command or Action	Purpose
Step 5	**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 6	**label mode** [**explicit-null** | **all-explicit-null**] **Example:** `Device(config-router-af)# label mode explicit-null`	Configures the IPv6 Explicit Null label • **explicit-null** label: for directly connected IPv6 prefixes sent to BGP labelled unicast neighbors. • **all-explicit-null** label: for all IPv6 prefixes sent to BGP labelled unciast neighbors.
Step 7	**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 8	**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 9	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits BGP address-family submode.
Step 10	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Configuration Examples for 6PE

Figure 5: 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)
```

Configuration Examples for IPV6 Explicit Null label for 6PE

The following example shows how to configure the IPv6 explicit-null label.

```
Device(config)# router bgp 1
Device(config-router)# address-family ipv6
Device(config-router-af)#label mode explicit-null
Device(config-router-af)# neighbor 33.33.33.33 activate
Device(config-router-af)#  neighbor 33.33.33.33 send-label
```

The following example shows how to configure the IPv6 all-explicit-null label.

```
Device(config)# router bgp 1
Device(config-router)# address-family ipv6
Device(config-router-af)#label mode all-explicit-null
Device(config-router-af)# neighbor 33.33.33.33 activate
Device(config-router-af)#  neighbor 33.33.33.33 send-label
```

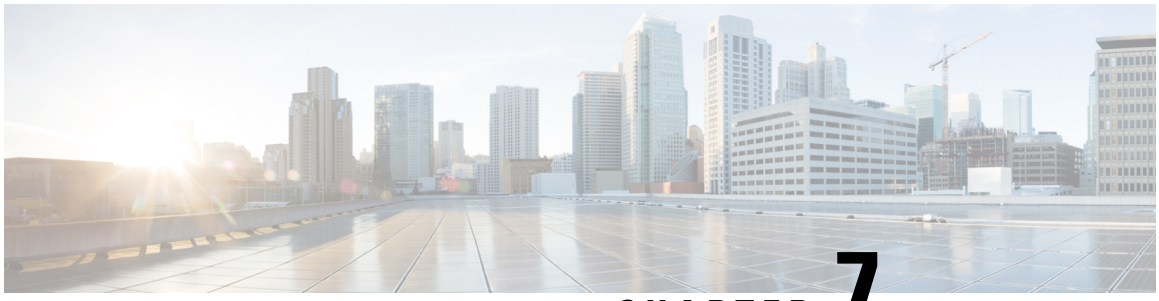
Feature History for IPv6 Provider Edge over MPLS (6PE)

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 Bengaluru 17.6.1	IPV6 Explicit Null label for 6PE	IPv6 Explicit Null Label is supported as a VPN label to exchange IPv6 reachability information over the MPLS core
Cisco IOS XE Cupertino 17.7.1	IPv6 Provider Edge over MPLS (6PE)	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 IPv6 VPN Provider Edge over MPLS (6VPE)

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

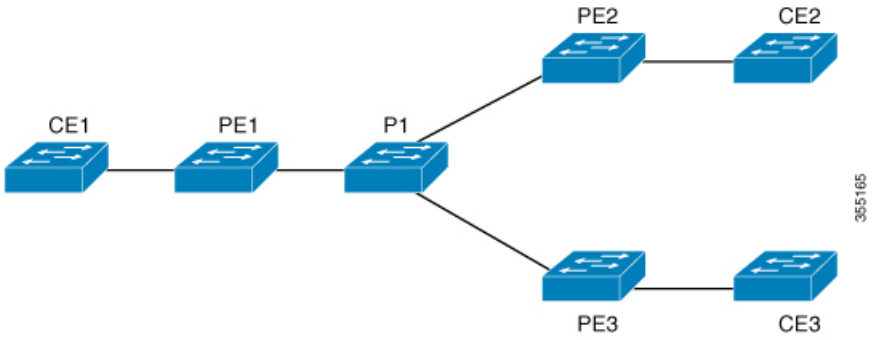
- VPN route target communities – A list of all other members of a VPN community.
- Multiprotocol BGP (MP-BGP) peering of VPN community PE routers – Propagates VRF reachability information to all members of a VPN community.

- 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 6: 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
```

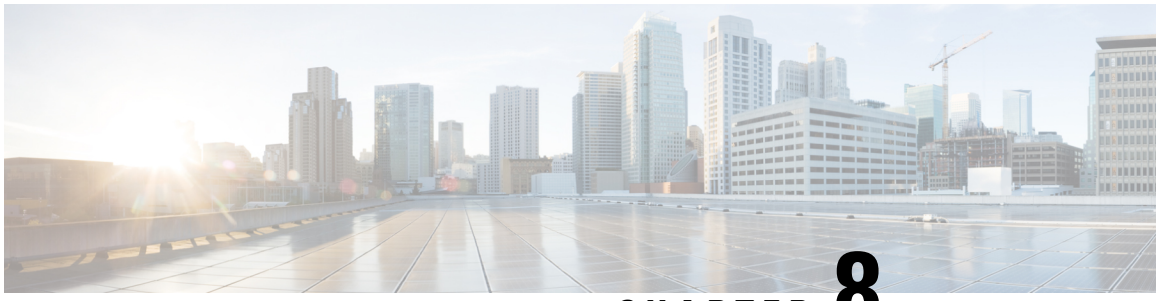
Feature History for IPv6 VPN Provider Edge over MPLS (6VPE)

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.
Cisco IOS XE Cupertino 17.7.1	IPv6 VPN Provider Edge over MPLS (6VPE)	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 VPN InterAS Options

Information About MPLS VPN InterAS Options

The MPLS VPN InterAS Options feature provides 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.

Autonomous Systems 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 autonomous system boundary router (ASBR) is a device in an AS that is configured 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 autonomous systems 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 autonomous systems, 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 autonomous systems. The primary function of eBGP is to exchange network reachability information between autonomous systems, including information about the list of AS routes. The autonomous systems 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 autonomous systems, connected by separate border edge routers. The autonomous systems exchange routes using eBGP, and no iBGP or routing information is exchanged between the autonomous systems.

MPLS VPN InterAS Options

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

- 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.
- InterAS Option AB: This option combines the best functionality of an interAS option A and interAS option B network to allow an MPLS VPN service provider to interconnect different autonomous systems to provide VPN services.

InterAS Option A

In terms of configuration, interAS Option A is the simplest of all available options.

A typical AS consists of these devices – Provider Edge(PE), Customer Edge(CE) and an Autonomous System Boundary Router(ASBR). The target is to enable VRF connectivity between CE devices (also referred to as VPN sites) in a network. In order to facilitate interAS option A, you have to perform the following for each VPN site:

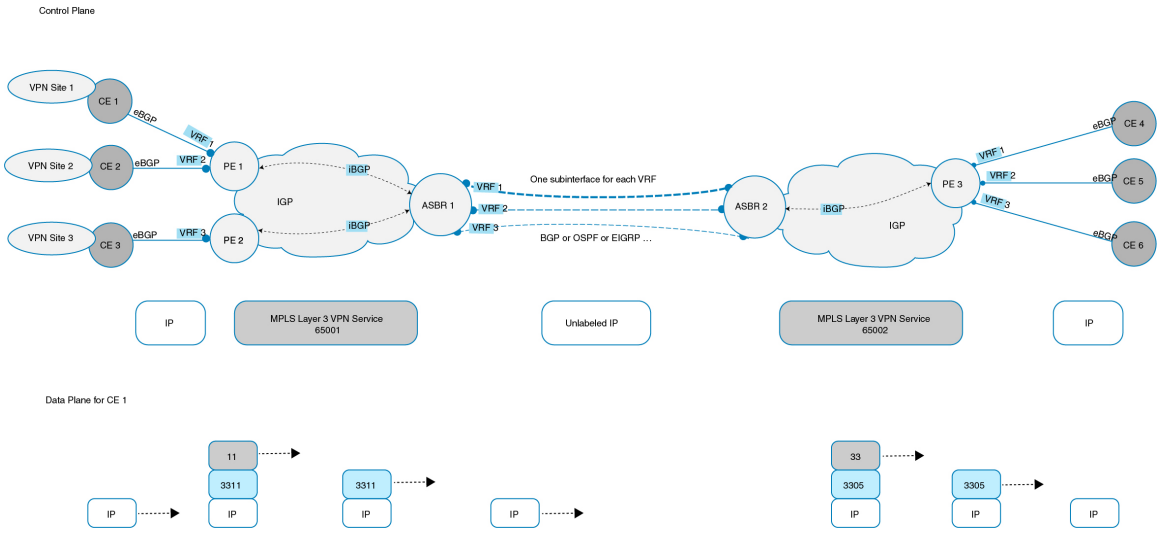
- Assign a VRF interface to each VPN site
- Define an interface or sub-interface for each VRF interface. (If multiple VPN sites are involved, they cannot all be associated with a single interface, and therefore, a sub-interface must be configured for each VRF). Optionally, a dedicated QoS policy may be applied to each subinterface.
- Create a BGP (or other routing protocol) session for each VRF.

With the above configuration in place, traffic flow with option A is as follows: Within the AS, data packets travel like regular Layer 3 VPN traffic. Traffic flow between ASBRs when traversing autonomous systems is in the form of unlabeled IP packets on a VRF interface. Any routing protocol may be used to exchange routing information between the ASBRs in the different autonomous systems.

While this option provides certain advantages (flexibility in terms of the routing protocol that can be used within an AS and between ASBRs, and security by means of a QoS policy on a subinterface), the scale for interAS option A is limited by the scale numbers for subinterfaces and VRFs. This option is therefore suited only to scenarios where the number of VPNs and the number of routes to transfer, is limited (and not likely to increase).

The figure below shows the data packet flow from CE 1, CE 2, CE 3 to CE 4, CE 5, CE 6 respectively. The explanation below takes the instance of the route advertisement and data packet flow from CE1 in AS-65001 to CE 4 in AS-65002.

Figure 7: MPLS VPN InterAS Option A Topology

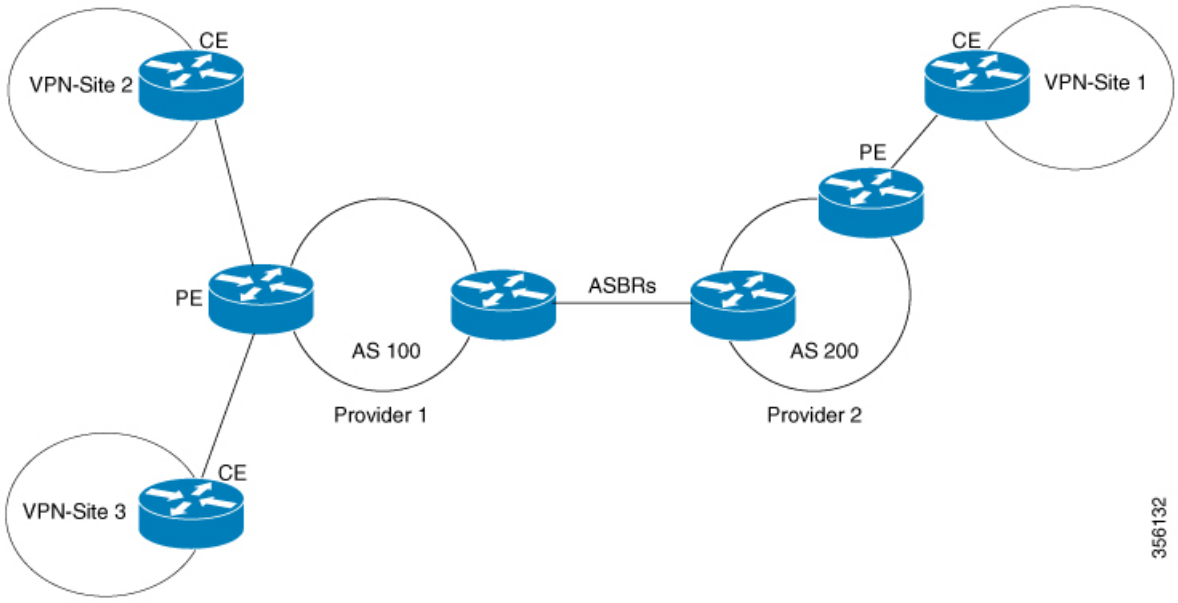


The IP traffic between CE 1 and PE 1 is sent over a VRF sub-interface by using eBGP. Once the packet reaches PE 1 it is sent to ASBR 1 as a two-label MPLS stack. The outermost label is the Interior Gateway protocol (IGP) label and the inner label is the VPN label. Layer 3 VPN traffic is sent from PE 1 to ASBR 1 in AS-65001 and from ASBR 2 to PE 3 in AS-65002 over a MPLS cloud. At ASBR 1, both the labels (IGP and VPN) are popped (removed). From ASBR 1 to ASBR 2 traffic flows as an unlabelled IP packet on a VRF interface. In this example, the routing protocol used between the two ASBRs is eBGP. The two label MPLS stack is pushed once the IP packet reaches ASBR 2. After the packet reaches PE 3, the VPN label is removed. The IGP label is also popped in case of explicit NULL IGP. The VPN packet is sent to CE4 through a VRF interface.

InterAS Option B

In an interAS option B network, ASBR ports are connected by one or more interfaces that are enabled to receive MPLS traffic. With this option, the ASBRs peer with each other using eBGP session. The ASBR also functions as a PE router and peers with every PE router in their AS. The ASBR does not hold any VRFs but holds all or a subset of VPNv4 routes from PE router that need to be passed to the other AS. VPNv4 routes are kept unique in ASBR using route-distinguisher and are filtered using route targets. The ASBRs exchange VPNv4 routes and VPN labels using eBGP.

Figure 8: Topology for InterAS Option B



Two methods are supported to distribute the next hop for VPNv4 routes between ASBRs. There is no requirement for LDP or any IGP to be enabled on the link connecting the two ASBRs. The MP-eBGP session between directly connected interfaces on the ASBRs enables the interfaces to forward labeled packets. To ensure this MPLS forwarding for directly connected BGP peers, you must configure mpls bgp forwarding command on the interface connecting to ASBR. This command is implemented in the IOS for directly connected interfaces. Upto 200 BGP neighbors can be configured.

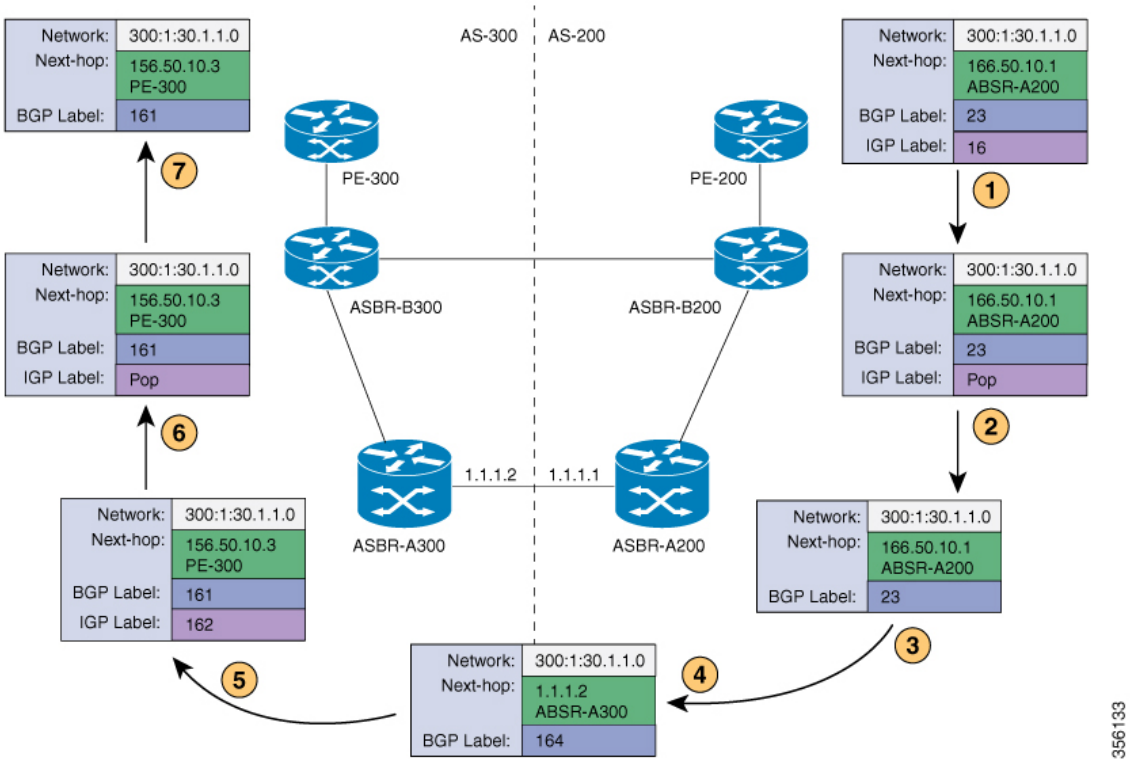
- **Next-hop-self Method:** Changing next-hop to that of the local ASBR for all VPNv4 routes learnt from the other ASBR.
- **Redistribute Connected Subnets Method:** Redistributing the next hop address of the remote ASBR into the local IGP using redistribute connected subnets command , i.e., the next hop is not changed when the VPNv4 routes are redistributed into the local AS.

Note In case of multiple equal paths - ECMP towards remote AS, you have to configure MPLS static label bindings towards remote Loopback on ASBR. Otherwise, you may experience packet loss.

The label switch path forwarding sections described below has AS200 configured with the Next-hop-self method and the AS300 is configured with Redistribute-subnet method.

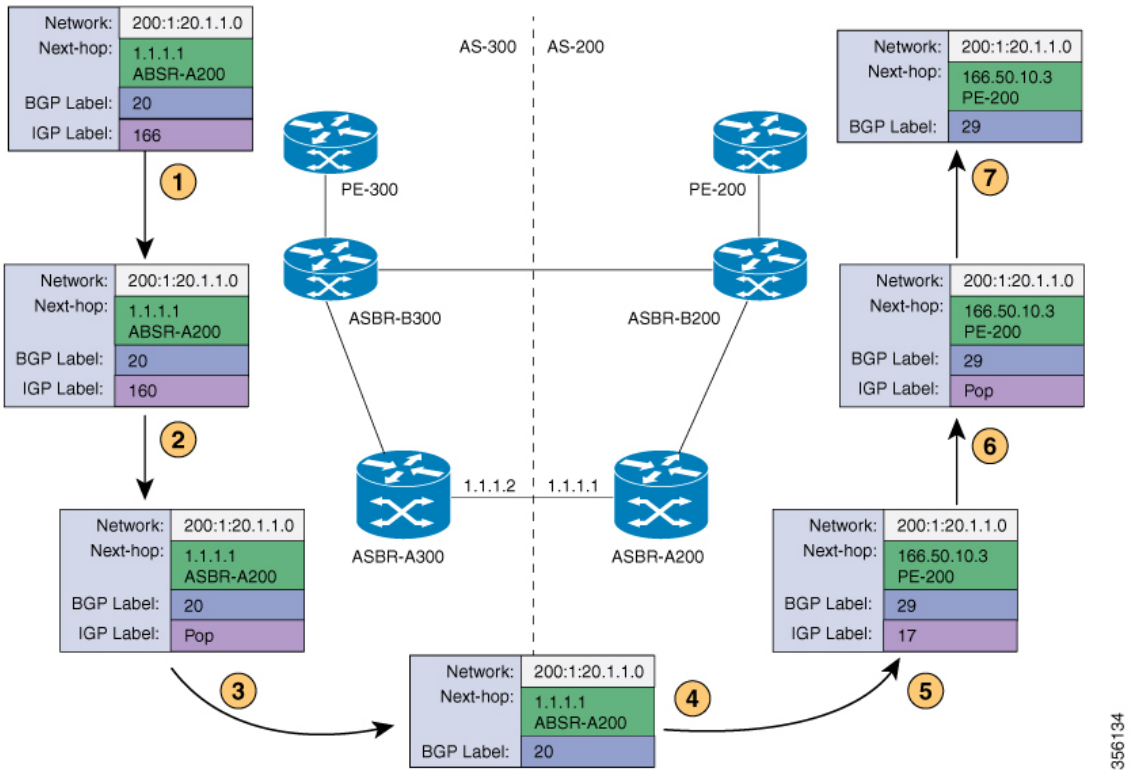
Next-Hop Self Method

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.



InterAS Option AB

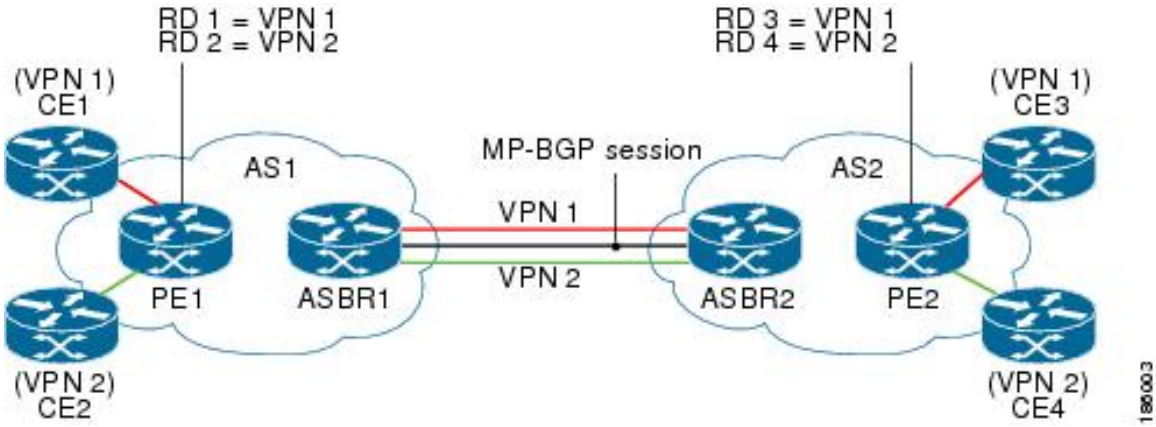
MPLS VPN service providers need to interconnect different autonomous systems to provide service for multiple VPN customers. The MPLS VPN InterAS Option AB feature allows the different autonomous systems to interconnect by using a single MP-BGP session in the global routing table to carry control plane traffic. This MP-BGP session signals VPN prefixes between two ASBRs for each VRF instance. This traffic can either be IP or MPLS.

MPLS BGP forwarding or LDP does not have to be configured between the two ASBRs because the VPN traffic that is IP traffic over a VRF-specific interface.

The interAS option AB feature provides the following benefits for service providers:

- IP QoS functions between ASBR peers are maintained for customer SLAs.
- Dataplane traffic is isolated on a per-VRF basis for security purposes.
- A dedicated QoS policy can be applied on each VRF by attaching the policy on an SVI.

Route Distribution and Packet Forwarding



The following attributes describe the topology of the sample interAS Option AB network shown in the figure above:

- CE1 and CE3 belong to VPN 1.
- CE2 and CE 4 belong to VPN 2.
- PE1 uses route distinguisher 1 (RD 1) for VPN 1 (VRF 1) and RD 2 for VPN 2 (VRF 2).
- PE2 uses RD 3 for VPN 1 (VRF 1) and RD 4 for VPN 2 (VRF 2).
- ASBR1 has VRF 1 provisioned with RD 5 and VRF 2 provisioned with RD 6.
- ASBR2 has VRF 1 provisioned with RD 7 and VRF 2 provisioned and RD 8.
- ASBR1 and ASBR2 have three links between them:
  - VRF 1
  - VRF 2
  - MP-BGP session

Route Distribution for VPN 1

A route distinguisher (RD) is an identifier attached to a route that identifies which VPN belongs to each route. Each routing instance must have a unique RD autonomous system associated with it. The RD is used to place a boundary around a VPN so that the same IP address prefixes can be used in different VPNs without having these IP address prefixes overlap. An RD statement is required if the instance type is a VRF.

The following process describes the route distribution process for VPN 1 in the figure above. Prefix “N” is used in this process to indicate the IP address of a VPN.

ASBR 1

- CE1 advertises the prefix N to PE1.
- PE1 advertises a VPN prefix RD 1:N to ASBR1 through MP-iBGP.
- ASBR1 imports the prefix into VPN 1 and creates a prefix RD 5:N.

- ASBR1 advertises the imported prefix RD 5:N to ASBR2. ASBR1 sets itself as the next hop for prefix RD 5:N and allocates a local label that is signaled with this prefix.
- ASBR1 advertises the route with the export RT configured on the VRF rather than the originally received RTs. By default, ASBR1 does not advertise the source prefix RD 1:N to ASBR2. This advertisement is suppressed because the prefix is being imported into an Option AB VRF.

ASBR 2

- ASBR2 receives the prefix RD 5:N and imports it into VPN 1 as RD 7:N.
- ASBR2 advertises the route with the export RT configured on the VRF rather than the originally received RTs.
- While importing the prefix, ASBR2 sets the next hop of RD 7:N to the ASBR1 interface IP address in VRF 1. The next hop table ID is also set to VRF 1. When installing the MPLS forwarding entry for RD 7:N, by default ASBR2 does not install the outgoing label in the forwarding process. This enables the traffic between the ASBRs to be IP.
- ASBR2 advertises the imported prefix RD 7:N to PE2. It sets itself as the next hop for this prefix and also allocates a local label that is signalled with the prefix. By default, ASBR2 does not advertise the source prefix RD 5:N to PE2. This advertisement is suppressed because the prefix is being imported into an Option AB VRF.
- PE2 imports the RD 7:N into VRF 1 as RD 3:N.

Packet Forwarding for VPN 1

The following packet forwarding process works the same as it does in an Option A scenario. The ASBR acts like the PE by terminating the VPN and then forwards its traffic as standard IP packets with no VPN label to the next PE, which in turn repeats the VPN process. Each PE device, therefore, treats the adjacent PE device as a CE device, and the standard Layer 3 MPLS VPN mechanisms are used for route redistribution with each autonomous system; that is, the PEs use external BGP (eBGP) to distribute unlabeled IPv4 addresses to each other.

- CE3 sends a packet destined for N to PE2.
- PE2 encapsulates the packet with the VPN label allocated by ASBR2 and the Interior Gateway Protocol (IGP) label needed to tunnel the packet to ASBR2.
- The packet arrives on ASBR2 with the VPN label. ASBR2 removes the VPN label and sends the packet as IP to ASBR1 on the VRF 1 interface.
- The IP packet arrives at ASBR1 on the VRF 1 interface. ASBR1 then encapsulates the packet with the VPN label allocated by PE1 and the IGP label needed to tunnel the packet to PE1.
- The packet arrives on PE1 with the VPN label. PE1 disposes the VPN label and forwards the IP packet to CE1.

Route Distribution for VPN 2

The following information describes the route distribution process for VPN 2 in the figure above:

ASBR 1

- CE2 advertises prefix N to PE1, where N is the VPN IP address.

- PE1 advertises a VPN prefix RD 2:N to ASBR1 through MP-iBGP.
- ASBR1 imports the prefix into VPN 2 and creates a prefix RD 6:N.
- ASBR1 advertises the imported prefix RD 6:N to ASBR2. It sets itself as the next hop for this prefix and also allocates a local label that is signalled with the prefix. By default, ASBR1 does not advertise the source prefix RD 2:N to ASBR2. This advertisement is suppressed as the prefix is being imported into an Option AB VRF.

ASBR 2

- ASBR2 receives the prefix RD 6:N and imports it into VPN 2 as RD 8:N.
- While importing the prefix, ASBR2 sets the next hop of RD 8:N to ASBR1s interface address in VRF 2. The next hop table ID is also set to that of VRF 2. While installing the MPLS forwarding entry for RD 8:N, by default ASBR2 does not install the outgoing label in the forwarding process. This enables traffic between the ASBRs to be IP.
- ASBR2 advertises the imported prefix RD 8:N to PE2. It sets itself as the next hop for this prefix and also allocates a local label that is signalled with the prefix. By default, ASBR2 does not advertise the source prefix RD 6:N to PE2. This advertisement is suppressed because the prefix is being imported into an Option AB VRF.
- PE2 imports the RD 8:N into VRF 2 as RD 4:N.

How to Configure MPLS VPN InterAS Options

The following section provides information about how to configure MPLS VPN InterAS Options.

Configuring MPLS VPN InterAS Option A

Sending AS: Configuring PE

Complete the following tasks to configure the PE which is in the AS sending data to another AS.

Sending AS: Configuring a VRF for a PE

Beginning in user EXEC mode complete the following steps to configure a VRF for a PE which is in the sending AS:

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	**vrf definition** *vrf-name*
	Example:
	`Device(config)# vrf definition cu1`
	`Device(config-vrf)#`
		Configures a VRF table and enters VRF configuration mode.

Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 1:1`	Creates routing and forwarding tables for a VRF instance.

Step 5	**address-family ipv4**
	Example:
	`Device(config-vrf)# address-family ipv4`
	`Device(config-vrf-af)#`
		Places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv6 address prefixes.

Step 6	**route-target export** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target export 100:1`	Creates a list of export route target communities for the specified VRF.
Step 7	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 100:2`	Creates a list of import route target communities for the specified VRF.

Step 8	**exit-address-family**
	Example:
	`Device(config-vrf-af)# exit-address-family`
	`Device(config-vrf)#`
		Exits the address family configuration mode and returns to VRF configuration mode.

Step 9	**address-family ipv6** **Example:** `Device(config-vrf)# address-family ipv6`	Places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv6 address prefixes.

	Command or Action	Purpose
Step 10	**route-target export** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target export 100:101`	Creates a list of export route target communities for the specified VRF.
Step 11	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 100:102`	Creates a list of import route target communities for the specified VRF.

Step 12	**exit-address-family**
	Example:
	`Device(config-vrf-af)# exit-address-family`
	`Device(config-vrf)#`
		Exits the address family configuration mode and returns to VRF configuration mode.

Sending AS: Configuring a PE-CE Interface

Beginning in privileged EXEC mode complete the following steps to configure a PE-CE interface which is in the sending AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface** {*interface-id* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface Gi1/1/0/13.1`
	`Device(config-if)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 3	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-if)# encapsulation dot1q 900`	Enables IEEE 802.1Q encapsulation of traffic on a specified interface.

	Command or Action	Purpose
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding cu1`	Associates the VRF with the Layer 3 interface.
Step 5	**ip address** *ip address mask* [**secondary**] **Example:** `Device(config-if)# ip address 140.1.1.1 255.255.255.0`	Sets a primary or secondary IP address for an interface.

Step 6	**exit**
	Example:
	`Device(config-if)# exit`
	`Device(config)#`
		Exits interface configuration mode and returns to global configuration mode.

Sending AS: Configuring BGP

Beginning in user EXEC mode complete the following steps to configure a BGP session for a PE which is in the sending AS:

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 65001`
	`Device(config-router)#`
		Configures a BGP routing process.

Step 4	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 2.2.2.2 remote-as 65001`	Configures an entry to the BGP neighbor table.

	Command or Action	Purpose

Step 5	**address-family** *ipv4* [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | [**vrf** *vrf-name*]
	Example:
	`Device(config-router)# address-family ipv4`
	`Device(config-router-af)#`
		Enters address family configuration mode for configuring BGP routing sessions that use standard IPv4 address prefixes.

Step 6	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 2.2.2.2 activate`	Enables the exchange of information with a BGP neighbor.

Step 7	**exit address-family**
	Example:
	`Device(config-router-af)# exit address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 8	**address-family** *vpnv4*
	Example:
	`Device(config-router)# address-family vpnv4`
	`Device(config-router-af)#`
		Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.

Step 9	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 2.2.2.2 activate`	Enables the exchange of information with a BGP neighbor.
Step 10	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send-community** [**both** | **standard** | **extended**] **Example:** `Device(config-router-af)# neighbor 2.2.2.2 send-community both`	Enables the exchange of information with a BGP neighbor.
Step 11	**exit address-family** **Example:** `Device(config-router-af)# exit`	Exits BGP address-family submode.

	Command or Action	Purpose
	address-family Device(config-router)#	
Step 12	**address-family** *vpnv6* **Example:** Device(config-router)# **address-family vpnv6** Device(config-router-af)#	Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv6 address prefixes.
Step 13	**neighbor** *ip-address* **activate** **Example:** Device(config-router-af)# **neighbor 2.2.2.2 activate**	Enables the exchange of information with a BGP neighbor.
Step 14	**neighbor** *ip-address* **send-community extended** **Example:** Device(config-router-af)# **neighbor 2.2.2.2 send-community extended**	Specifies that a community attribute should be sent to a BGP neighbor.
Step 15	**exit address-family** **Example:** Device(config-router-af)# **exit address-family** Device(config-router)#	Exits BGP address-family submode.
Step 16	**address-family ipv4 vrf** *vrf-name* **Example:** Device(config-router)# **address-family ipv4 vrf cu1** Device(config-router-af)#	Enters address family configuration mode for configuring BGP routing sessions that use standard IPv4 address prefixes.
Step 17	**redistribute** *protocol* **Example:** Device(config-router-af)# **redistribute connected**	Redistributes routes from one routing domain into another routing domain.
Step 18	**neighbor** *ip-address* **remote-as** *as-number* **Example:** Device(config-router-af)# **neighbor 140.1.1.2 remote-as 65002**	Configures an entry to the BGP neighbor table.

	Command or Action	Purpose
Step 19	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 140.1.1.2 activate`	Enables the exchange of information with a BGP neighbor.

Step 20	**exit address-family**
	Example:
	`Device(config-router-af)# exit address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 21	**exit** **Example:** `Device(config-router)# exit`	Exits router BGP mode.

Sending AS: Configuring a PE-P Interface and IGP

Beginning in user EXEC mode complete the following steps to configure a PE-P interface and IGP which is in the sending AS:

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* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface po91`
	`Device(config-if)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Sets the interface to the routed-interface status and erases all Layer 2 configurations.

	Command or Action	Purpose
Step 5	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 91.1.1.1 255.255.255.248`	Sets a primary or secondary IP address for an interface.
Step 6	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 2 area 0`	Enables OSPF on an interface.
Step 7	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 9	**router ospf** *process-id* **Example:** `Device(config)# router ospf 2`	Configures an OSPF routing process and assigns a process number.
Step 10	**router-id** *ip-address* **Example:** `Device(config-router)# router-id 1.1.1.1`	Specifies a fixed router ID.
Step 11	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Sending AS: Configuring P

Complete the following tasks to configure the P which is in the AS sending data to another AS.

Sending AS: Configuring P-PE Interface and IGP

Beginning in user EXEC mode complete the following steps to configure a P-PE interface and IGP which is in the sending AS:

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	**interface** {*interface-id* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface Port-channel91`
	`Device(config-if)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Sets the interface to the routed-interface status and erases all Layer 2 configuration.
Step 5	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 91.1.1.2 255.255.255.248`	Sets a primary or secondary IP address for an interface.
Step 6	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 2 area 0`	Enables OSPF on an interface.
Step 7	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.

Step 8	**exit**
	Example:
	`Device(config-if)# exit`
	`Device(config)#`
		Exits interface configuration mode.

Step 9	**interface** {*interface-id* | *subinterface-id* | *vlan-id*} **Example:** `Device(config)# interface Port-channel92`	Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.
Step 10	**no switchport** **Example:**	Set the interface to the routed-interface status erases all Layer 2 configurations.

	Command or Action	Purpose
	`Device(config-if)# no switchport`	
Step 11	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 92.1.1.2 255.255.255.248`	Sets a primary or secondary IP address for an interface.
Step 12	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 2 area 0`	Enables OSPF on an interface.
Step 13	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.
Step 14	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Step 15	**router ospf** *process-id*
	Example:
	`Device(config)# router ospf 2`
	`Device(config-router)#`
		Configures an OSPF routing process and assign a process number.

Step 16	**router-id** *ip-address* **Example:** `Device(config-router)# router-id 5.5.5.5`	Specifies a fixed router ID.
Step 17	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode, and returns to privileged EXEC mode.

Sending AS: Configuring ASBR

Complete the following tasks to configure the ASBR which is in the AS sending data to another AS.

Sending AS: Configuring VRF for ASBR

Beginning in user EXEC mode complete the following steps to configure a VRF for a ASBR which is in the sending AS:

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	**vrf definition** *vrf-name*
	Example:
	`Device(config)# vrf definition cu1`
	`Device(config-vrf)#`
		Configures a VRF table and enters VRF configuration mode.

Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 1:2`	Creates routing and forwarding tables for a VRF instance.

Step 5	**address-family ipv4**
	Example:
	`Device(config-vrf)# address-family ipv4`
	`Device(config-vrf-af)#`
		The address-family ipv4 command places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv4 address prefixes.

Step 6	**route-target export** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target export 100:2`	Creates a list of export route target communities for the specified VRF.
Step 7	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 100:1`	Creates a list of import route target communities for the specified VRF.
Step 8	**exit-address-family** **Example:** `Device(config-vrf-af)#`	Leaves the address family configuration mode and returns to router configuration mode.

	Command or Action	Purpose
	exit-address-family Device(config-vrf)#	
Step 9	**address-family ipv6** **Example:** Device(config-vrf)# **address-family ipv6**	Places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv6 address prefixes.
Step 10	**route-target export** *route-target-ext-community* **Example:** Device(config-vrf-af)# **route-target export 100:102**	Creates a list of export route target communities for the specified VRF.
Step 11	**route-target import***route-target-ext-community* **Example:** Device(config-vrf-af)# **route-target import 100:101**	Creates a list of import route target communities for the specified VRF.
Step 12	**exit-address-family** **Example:** Device(config-vrf-af)# **exit-address-family** Device(config-vrf)#	Exits the address family configuration mode and returns to router configuration mode.
Step 13	**exit** **Example:** Device(config-vrf)# **exit**	Exits the router configuration mode and returns to global configuration mode.

Sending AS: Configuring Interface Towards the Receiving ASBR

Beginning in privileged EXEC mode complete the following steps to configure an interface towards the receiving ASBR:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** Device# **configure terminal**	Enters global configuration mode.

	Command or Action	Purpose

Step 2	**interface** {*interface-id* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface fo1/0/10.1`
	`Device(config-subif)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 3	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 900`	Enables IEEE 802.1Q encapsulation of traffic on a specified interface.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-subif)# vrf forwarding cu1`	Associates the VRF with the Layer 3 interface.
Step 5	**ip address** *ip address mask* [**secondary**] **Example:** `Device(config-subif)# ip address 141.1.1.1 255.255.255.0`	Sets a primary or secondary IP address for an interface.

Sending AS: Configuring BGP

Beginning in privileged EXEC mode complete the following steps to configure a BGP session on the ASBR which is in the sending AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `Device(config-if)# router bgp 65001`	Configures a BGP routing process.
Step 3	**bgp log-neighbor changes** **Example:**	Enables logging of BGP neighbor resets.

	Command or Action	Purpose
	`Device(config-router)# bgp log-neighbor-changes`	
Step 4	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 1.1.1.1 remote-as 65001`	Configures an entry to the BGP neighbor table.
Step 5	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 1.1.1.1 update-source Loopback0`	Allows Cisco IOS software to use a specific operational interface for TCP connections by the BGP sessions.

Step 6	**address-family ipv4** [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | [**vrf** *vrf-name*]
	Example:
	`Device(config-router)# address-family ipv4`
	`Device(config-router-af)#`
		Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.

Step 7	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 1.1.1.1 activate`	Enables the exchange of information with a BGP neighbor.
Step 8	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits BGP address-family submode.
Step 9	**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 10	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 1.1.1.1 activate`	Enables the exchange of information with a BGP neighbor.

	Command or Action	Purpose
Step 11	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send-community** [**both** | **standard** | **extended**] **Example:** `Device(config-router-af)# neighbor 1.1.1.1 send-community both`	Enables the exchange of information with a BGP neighbor.

Step 12	**exit-address-family**
	Example:
	`Device(config-router-af)# exit-address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 13	**address-family** *vpnv6*
	Example:
	`Device(config-router)# address-family vpnv6`
	`Device(config-router-af)#`
		Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv6 address prefixes.

Step 14	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 1.1.1.1 activate`	Enables the exchange of information with a BGP neighbor.
Step 15	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send-community** [**both** | **standard** | **extended**] **Example:** `Device(config-router-af)# neighbor 1.1.1.1 send-community both`	Enables the exchange of information with a BGP neighbor.

Step 16	**exit-address-family**
	Example:
	`Device(config-router-af)# exit-address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

	Command or Action	Purpose
Step 17	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv4 vrf cu1`	Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.
Step 18	**redistribute** *protocol* **Example:** `Device(config-router-af)# redistribute connected`	Redistributes routes from one routing domain into another routing domain.
Step 19	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router-af)# neighbor 141.1.1.2 remote-as 65002`	Configures an entry to the BGP neighbor table.
Step 20	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 141.1.1.2 activate`	Enables the exchange of information with a BGP neighbor.
Step 21	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits BGP address-family submode.

Sending AS: Configuring a ASBR-P Interface and a IGP

Beginning in privileged EXEC mode complete the following steps to configure a ASBR-P interface and a IGP in the sending AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** {*interface-id* | *subinterface-id* | *vlan-id*} **Example:**	Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

	Command or Action	Purpose
	`Device(config)# interface Port-channel92`	
Step 3	**no switchport** **Example:** `Device(config-if)# no switchport`	Set the interface to the routed-interface status erases all Layer 2 configurations.
Step 4	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 92.1.1.1 255.255.255.248`	Sets a primary or secondary IP address for an interface.
Step 5	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 2 area 0`	Enables OSPF on an interface.
Step 6	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables Multiprotocol Label Switching (MPLS) forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Receiving AS: Configuring ASBR

Complete the following tasks to configure the ASBR which is in the AS receiving data from another AS.

Receiving AS: Configuring VRF for ASBR

Beginning in user EXEC mode complete the following steps to configure a VRF for a ASBR which is in the receiving AS:

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	**vrf definition** *vrf-name*
	Example:
	`Device(config)# vrf definition cu1`
	`Device(config-vrf)#`
		Configures a VRF table and enters VRF configuration mode.

Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 1:3`	Creates routing and forwarding tables for a VRF instance.

Step 5	**address-family ipv4**
	Example:
	`Device(config-vrf)# address-family ipv4`
	`Device(config-vrf-af)#`
		The address-family ipv4 command places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv4 address prefixes.

Step 6	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 200:2`	Creates a list of export route target communities for the specified VRF.
Step 7	**route-target export** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target export 200:1`	Creates a list of import route target communities for the specified VRF.
Step 8	**exit-address-family** **Example:** `Device(config-vrf-af)# exit-address-family`	Leaves the address family configuration mode and returns to router configuration mode.

Step 9	**address-family ipv6**
	Example:
	`Device(config-vrf)# address-family ipv6`
	`Device(config-vrf-af)#`
		Places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv6 address prefixes.

Step 10	**route-target export** *route-target-ext-community* **Example:**	Creates a list of export route target communities for the specified VRF.

	Command or Action	Purpose
	`Device(config-vrf-af)# route-target export 200:101`	
Step 11	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 200:102`	Creates a list of import route target communities for the specified VRF.

Step 12	**exit-address-family**
	Example:
	`Device(config-vrf-af)# exit-address-family`
	`Device(config-vrf)#`
		Exits the address family configuration mode and returns to router configuration mode.

Step 13	**exit** **Example:** `Device(config-vrf)# exit`	Exits the router configuration mode and returns to global configuration mode.

Receiving AS: Configuring Interface Towards the Sending ASBR

Beginning in privileged EXEC mode complete the following steps to configure an interface towards the sending ASBR:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface** {*interface-id* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface fo1/0/10.1`
	`Device(config-subif)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 3	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 900`	Enables IEEE 802.1Q encapsulation of traffic on a specified interface.

	Command or Action	Purpose
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-subif)# vrf forwarding cu1`	Associates the VRF with the Layer 3 interface.
Step 5	**ip address** *ip address mask* [**secondary**] **Example:** `Device(config-subif)# ip address 141.1.1.1 255.255.255.0`	Sets a primary or secondary IP address for an interface.

Step 6	**exit**
	Example:
	`Device(config-subif)# exit`
	`Device(config)#`
		Exits to global configuration mode.

Receiving AS: Configuring BGP

Beginning in privileged EXEC mode complete the following steps to configure a BGP session on the ASBR which is in the receiving AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**router bgp** *autonomous-system-number*
	Example:
	`Device(config)# router bgp 65002`
	`Device(config-router)#`
		Configures a BGP routing process.

Step 3	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 30.30.30.30 remote-as 65002`	Configures an entry to the BGP neighbor table.
Step 4	**address-family** *ipv4* [**mdt** | **multicast** | **tunnel** | **unicast** [**vrf** *vrf-name*] | [**vrf** *vrf-name*] **Example:**	Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.

	Command or Action	Purpose
	`Device(config-router)# address-family ipv4` `Device(config-router-af)#`	
Step 5	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 30.30.30.30 activate`	Enables the exchange of information with a BGP neighbor.

Step 6	**exit**
	Example:
	`Device(config-router-af)# exit`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 7	**address-family** *ipv6*
	Example:
	`Device(config-router)# address-family ipv6`
	`Device(config-router-af)#`
		Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.

Step 8	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 30.30.30.30 activate`	Enables the exchange of information with a BGP neighbor.

Step 9	**exit address-family**
	Example:
	`Device(config-router-af)# exit address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 10	**address-family** *vpnv4*
	Example:
	`Device(config-router)# address-family vpnv4`
	`Device(config-router-af)#`
		Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv6 address prefixes.

Step 11	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 30.30.30.30 activate`	Enables the exchange of information with a BGP neighbor.

	Command or Action	Purpose
Step 12	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send-community** [**both** | **standard** | **extended**] **Example:** `Device(config-router-af)# neighbor 30.30.30.30 send-community both`	Enables the exchange of information with a BGP neighbor.

Step 13	**exit**
	Example:
	`Device(config-router-af)# exit`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 14	**address-family** *vpnv6*
	Example:
	`Device(config-router)# address-family vpnv6`
	`Device(config-router-af)#`
		Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.

Step 15	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 30.30.30.30 activate`	Enables the exchange of information with a BGP neighbor.
Step 16	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send-community** [**both** | **standard** | **extended**] **Example:** `Device(config-router-af)# neighbor 30.30.30.30 send-community both`	Enables the exchange of information with a BGP neighbor.

Step 17	**exit**
	Example:
	`Device(config-router-af)# exit`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 18	**address-family** *ipv4*
	Example:
	`Device(config-router)# address-family ipv4 vrf cu1`
	`Device(config-router-af)#`
		Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.

	Command or Action	Purpose
Step 19	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router-af)# neighbor 141.1.1.1 remote-as 65001`	Configures an entry to the BGP neighbor table.
Step 20	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 141.1.1.1 activate`	Enables the exchange of information with a BGP neighbor.

Step 21	**exit address-family**
	Example:
	`Device(config-router-af)# exit address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 22	**end** **Example:** `Device(config-router)# end`	Exits router BGP mode and returns to privileged EXEC mode.

Receiving AS: Configuring a ASBR-P Interface and a IGP

Beginning in privileged EXEC mode complete the following steps to configure a ASBR-P interface and a IGP which is in the receiving AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** {*interface-id* | *subinterface-id* | *vlan-id*} **Example:** `Device(config)# interface FortyGigabitEthernet1/0/13`	Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.
Step 3	**no switchport** **Example:**	Set the interface to the routed-interface status erases all Layer 2 configurations.

	Command or Action	Purpose
	`Device(config-if)# no switchport`	
Step 4	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 10.1.1.1 255.255.255.0`	Sets a primary or secondary IP address for an interface.
Step 5	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 10 area 0`	Enables OSPF on an interface.
Step 6	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables Multiprotocol Label Switching (MPLS) forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Receiving AS: Configuring P

Complete the following tasks to configure the P which is in the AS receiving data from another AS.

Receiving AS: Configuring ASBR-P Interface and IGP

Beginning in user EXEC mode complete the following steps to configure a ASBR-P interface and IGP which is in the receiving AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface**
	{*interface-id* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface HundredGigE1/0/13`
	`Device(config-if)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

	Command or Action	Purpose
Step 3	**no switchport** **Example:** `Device(config-if)# no switchport`	Set the interface to the routed-interface status erases all Layer 2 configurations.
Step 4	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 10.1.1.2 255.255.255.0`	Sets a primary or secondary IP address for an interface.
Step 5	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 10 area 0`	Enables OSPF on an interface.
Step 6	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables Multiprotocol Label Switching (MPLS) forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.
Step 7	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Step 8	**interface** { *interface-id* | *subinterface-id* | *vlan-id* }
	Example:
	`Device(config)# interface HundredGigE1/0/4`
	`Device(config-if)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 9	**no switchport** **Example:** `Device(config-if)# no switchport`	Set the interface to the routed-interface status and erases all Layer 2 configurations.
Step 10	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 20.1.1.1 255.255.255.0`	Sets a primary or secondary IP address for an interface.
Step 11	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 10 area 0`	Enables OSPF on an interface.
Step 12	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS forwarding of IPv4 and IPv6 packets along normally routed paths for a particular interface.

	Command or Action	Purpose

Step 13	**exit**
	Example:
	`Device(config-if)# exit`
	`Device(config)#`
		Exits interface configuration mode and returns to global configuration mode.

Step 14	**exit** **Example:** `Device(config)# exit`	Exits router configuration mode, and returns to privileged EXEC mode.

Receiving AS: Configuring PE

Complete the following tasks to configure the PE which is in the AS receiving data from another AS.

Configuring VRF for PE2

Beginning in privileged EXEC mode complete the following steps to configure a VRF for a PE:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**vrf definition** *vrf-name*
	Example:
	`Device(config)# vrf definition cu1`
	`Device(config-vrf)#`
		Configures a VRF table and enters VRF configuration mode.

Step 3	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 1:4`	Creates routing and forwarding tables for a VRF instance.

Step 4	**address-family ipv4**
	Example:
	`Device(config-vrf)# address-family ipv4`
	`Device(config-vrf-af)#`
		The address-family ipv4 command places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv4 address prefixes.

Step 5	**route-target export** *route-target-ext-community* **Example:**	Creates a list of export route target communities for the specified VRF.

	Command or Action	Purpose
	`Device(config-vrf-af)# route-target export 200:2`	
Step 6	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 200:1`	Creates a list of import route target communities for the specified VRF.

Step 7	**exit-address-family**
	Example:
	`Device(config-vrf-af)# exit-address-family`
	`Device(config-vrf)#`
		Leaves the address family configuration mode and returns to router configuration mode.

Step 8	**address-familyipv6**
	Example:
	`Device(config-vrf)# address-family ipv6`
	`Device(config-vrf-af)#`
		Places the device in address family configuration mode, from which you can configure routing sessions that use standard IPv6 address prefixes.

Step 9	**route-target export** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target export 200:102`	Creates a list of export route target communities for the specified VRF.
Step 10	**route-target import** *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 200:101`	Creates a list of import route target communities for the specified VRF.

Step 11	**exit-address-family**
	Example:
	`Device(config-vrf-af)# exit-address-family`
	`Device(config-vrf)#`
		Exits the address family configuration mode and returns to router configuration mode.

Step 12	**exit**
	Example:
	`Device(config-vrf)# exit`
	`Device(config)#`
		Exits the router configuration mode and returns to global configuration mode.

Receiving AS: Configuring PE-CE Interface

Beginning in privileged EXEC mode complete the following steps to configure a PE-CE interface which is in the receiving AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 2	**interface** {*interface-id* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface FortyGigabitEthernet1/0/5.1`
	`Device(config-subif)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 3	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 900`	Enables IEEE 802.1Q encapsulation of traffic on a specified interface.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-subif)# vrf forwarding cu1`	Associates the VRF with the Layer 3 interface.
Step 5	**ip address** *ip address mask* [**secondary**] **Example:** `Device(config-subif)# ip address 151.1.1.1 255.255.255.0`	Sets a primary or secondary IP address for an interface.

Step 6	**exit**
	Example:
	`Device(config-subif)# exit`
	`Device(config)#`
		Exits interface configuration mode and returns to global configuration mode.

Receiving AS: Configuring BGP

Beginning in privileged EXEC mode complete the following steps to configure a BGP session on a PE which is in the receiving AS:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `Device(config-if)# router bgp 65002`	Configures a BGP routing process.
Step 3	**bgp log-neighbor changes** **Example:** `Device(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 4	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 10.10.10.10 remote-as 65002`	Configures an entry to the BGP neighbor table.
Step 5	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router)# neighbor 10.10.10.10 update-source Loopback30`	Allows Cisco IOS software to use a specific operational interface for TCP connections by the BGP sessions.

Step 6	**address-family ipv4**
	Example:
	`Device(config-router)# address-family ipv4`
	`Device(config-router-af)#`
		Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.

Step 7	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.10.10.10 activate`	Enables the exchange of information with a BGP neighbor.
Step 8	**exit-address-family** **Example:**	Exits BGP address-family submode.

	Command or Action	Purpose
	`Device(config-router-af)# exit address-family` `Device(config-router)#`	

Step 9	**address-family** *vpnv4*
	Example:
	`Device(config-router)# address-family vpnv4`
	`Device(config-router-af)#`
		Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.

Step 10	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.10.10.10 activate`	Enables the exchange of information with a BGP neighbor.
Step 11	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send−community** [**both** | **standard** | **extended**] **Example:** `Device(config-router-af)# neighbor 10.10.10.10 send-community both`	Enables the exchange of information with a BGP neighbor.

Step 12	**exit-address-family**
	Example:
	`Device(config-router-af)# exit-address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 13	**address-family ipv6** **Example:** `Device(config-router)# address-family ipv6`	Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv6 address prefixes.
Step 14	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 10.10.10.10 activate`	Enables the exchange of information with a BGP neighbor.
Step 15	**exit-address-family** **Example:** `Device(config-router-af)#`	Exits BGP address-family submode.

	Command or Action	Purpose
	exit-address-family Device(config-router)#	
Step 16	**address-family** *vpnv6* **Example:** Device(config-router)# **address-family vpnv6**	Configures the device in address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.
Step 17	**neighbor** *ip-address* **activate** **Example:** Device(config-router-af)# **neighbor 10.10.10.10 activate**	Enables the exchange of information with a BGP neighbor.
Step 18	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **send-community** [**both** | **standard** | **extended**] **Example:** Device(config-router-af)# **neighbor 10.10.10.10 send-community both**	Enables the exchange of information with a BGP neighbor.
Step 19	**exit address-family** **Example:** Device(config-router-af)# **exit address-family**	Exits BGP address-family submode.
Step 20	**address-family ipv4 vrf** *vrf-name*] **Example:** Device(config-router)# **address-family ipv4 vrf cu1** Device(config-router-af)#	Enters address family configuration mode for configuring BGP routing sessions that use standard IP Version 4 address prefixes.
Step 21	**redistribute** *protocol* **Example:** Device(config-router-af)# **redistribute connected**	Redistributes routes from one routing domain into another routing domain.
Step 22	**neighbor** *ip-address* **remote-as** *as-number* **Example:** Device(config-router-af)# **neighbor 151.1.1.2 remote-as 65003**	Configures an entry to the BGP neighbor table.

	Command or Action	Purpose
Step 23	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 151.1.1.2 activate`	Enables the exchange of information with a BGP neighbor.

Step 24	**exit address-family**
	Example:
	`Device(config-router-af)# exit address-family`
	`Device(config-router)#`
		Exits BGP address-family submode.

Step 25	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode.

Receiving AS: Configuring a PE-P Interface and IGP

Beginning in user EXEC mode complete the following steps to configure a PE-P interface and IGP which is in the receiving AS:

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* | *subinterface-id* | *vlan-id*}
	Example:
	`Device(config)# interface FortyGigabitEthernet1/0/4`
	`(config-if)#`
		Enters interface configuration mode and specifies the Ethernet interface, subinterface, or VLAN to be associated with the VRF.

Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Set the interface to the routed-interface status erases all Layer 2 configurations.

	Command or Action	Purpose
Step 5	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 20.1.1.2 255.255.255.0`	Sets a primary or secondary IP address for an interface.
Step 6	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip ospf 10 area 0`	Enables OSPF on an interface.

Step 7	**end**
	Example:
	`Device(config-if)# end`
	`Device(config)#`
		Exits interface configuration mode and returns to privileged EXEC mode.

Configuring MPLS VPN InterAS Option B

The following section provides information about how to configure interAS option B using next-hop-self method and redistribute connected method.

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:** `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:** `Device(config)# router-id 4.1.1.1`	Specifies a fixed router ID.

	Command or Action	Purpose
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.
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.

	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 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.
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.

	Command or Action	Purpose
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.
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.

	Command or Action	Purpose
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.
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.

	Command or Action	Purpose
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:** `Device(config)# router-id 5.1.1.1`	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).

	Command or Action	Purpose
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 0.0.0.255 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.
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.

	Command or Action	Purpose
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.
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.

	Command or Action	Purpose
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.

Configuring MPLS VPN Inter-AS Option AB

The following sections describe how to configure the interAS option AB feature on an ASBR for an MPLS VPN:

Configuring the VRFs on the ASBR Interface for Each VPN Customer

Use the following steps to configure the VRFs on the ASBR interface for each VPN customer so that these VPNs have connectivity over the interAS Option AB network.

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`	Specifies the interface to configure and enters interface configuration mode.

	Command or Action	Purpose
	`1/0/1`	
Step 4	**ip vrf forwarding***vrf-name* **Example:** `Device(config-if)# ip vrf forwarding vpn1`	Associates a VRF with the specified interface. • The vrf-name argument is the name assigned to a VRF.
Step 5	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.

Configuring the MP-BGP Session Between ASBR Peers

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

Follow the steps in this section to configure the MP-BGP session on the ASBR.

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 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. Valid numbers are from 0 to 65535. Private autonomous system numbers that can be used in internal networks range from 64512 to 65535.

	Command or Action	Purpose
Step 4	**neighbor** { *ip-address* | *peer-group-name* } **remote-as** *as-number* **Example:** `Device(config-router)# neighbor 192.168.0.1 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 use standard VPNv4 address prefixes. • The unicast keyword specifies IPv4 unicast address prefixes.
Step 6	**neighbor** { *ip-address* | *peer-group-name* } **activate** **Example:** `Device(config-router-af)# neighbor 192.168.0.1 activate`	Enables the exchange of information with a neighboring 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* } **inter-as-hybrid** **Example:** `Device(config-router-af)# neighbor 192.168.0.1 inter-as-hybrid`	Configures eBGP peer device (ASBR) as an Inter-AS Option AB peer. • The *ip-address* argument specifies the IP address of the neighbor. • The *peer-group-name* argument specifies the name of a BGP peer group. • If any prefixes are imported into Option AB VRFs, then the imported paths are advertised to this peer. • If any prefixes are received from this peer and are imported into Option AB VRFs, then the imported paths are advertised to iBGP peers. **Note** Advertised routes have RTs that are configured on the VRF. Advertised routes do not have their original RTs.

	Command or Action	Purpose
Step 8	**exit-address-family** **Example:** `Device(config-router)# exit-address-family`	Exits from address family configuration mode.

Configuring the Routing Policy for VPNs that Need Inter-AS Connections

Use the steps in this section to configure VRFs for the VPNs that need Inter-AS connections between ASBR peers, by configuring the appropriate routing policy and Option AB 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	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vpn1`	Defines the VPN routing instance by assigning a 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 an RD in either of these formats:
		  • 16-bit autonomous system number: your 32-bit number, for example, 101:3.
		  • 32-bit IP address: your 16-bit number, for example, 192.168.122.15:1.

Step 5	**address-family ipv4** **Example:** `Device(config-vrf)# address-family ipv4`	Enters VRF address family configuration mode to specify an address family for a VRF. • The **ipv4** keyword specifies an IPv4 address family for a VRF.

	Command or Action	Purpose
		• 16-bit autonomous system number: your 32-bit number, for example, 101:3. • 32-bit IP address: your 16-bit number, for example, 192.168.122.15:1.
Step 6	**route-target** {**import** | **export** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf-af)# route-target import 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-targetextended community attributes to the VRF list of import, export, or both (import and export) route-target extended communities.
Step 7	**inter-as-hybrid** **Example:** `Device(config-vrf-af)# inter-as-hybrid`	Specifies the VRF as an option AB VRF, which has the following effects: • Routes imported to this VRF can be advertised to option AB peers and VPNv4 iBGP peers. • When routes received from option AB peers and are imported into the VRF, the next hop table ID of the route is set to the table ID of the VRF.
Step 8	**inter-as-hybrid** [**next-hop** *ip-address*] **Example:** `Device(config-vrf-af)# inter-as-hybrid next-hop 192.168.1.0`	(Optional) Specifies the next hop IP address to be set on paths that are imported into the VRF and that are received from an Option AB peer. • The next hop context is also set to the VRF, which imports these paths.
Step 9	**end** **Example:** `Device(config-vrf-af)# end`	(Optional) Exits to privileged EXEC mode.

Changing an Inter-AS Option A Deployment to an Option AB Deployment

In an option A deployment, the VRF instances are back-to-back between the ASBR devices and there is direct connectivity between PE devices of different autonomous systems. The PE devices are attached by multiple physical or logical interfaces, each of which is associated with a given VPN (through a VRF instance).

In the Option AB deployment, the different autonomous systems interconnect by using a single MP-BGP session in the global routing table to carry control plane traffic.

Use the following steps to change an MPLS VPN Inter-AS Option A deployment to an Option AB deployment.

1. Configure the MP-BGP session on the ASBR. BGP multiprotocol extensions are used to define support for address families other than IPv4 so that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.
2. Identify the VRFs that need an upgrade from Option A and configure them for Option AB by using the **inter-as-hybrid** command.
3. Use the following steps in this section to remove the configuration for the eBGP (peer ASBR) neighbor.
4. Repeat all the steps in the following procedure to remove the configuration for additional eBGP (peer ASBR) 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 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. Valid numbers are from 0 to 65535. Private autonomous system numbers that can be used in internal networks range from 64512 to 65535.
Step 4	**address-family ipv4 vrf** *vrf-name* **Example:**	Configures each VRF that is identified in the MP-BGP session on the ASBR so that the routes for a given VPN are learned only by other

	Command or Action	Purpose
	`Device(config-router)# address-family ipv4 vrf vpn4`	members of that VPN, enabling members of the VPN to communicate with each other. • Enters address family configuration mode to specify an address family for a VRF.
Step 5	**no neighbor** { *ip-address* | *peer-group-name* } **Example:** `Device(config-router-af)# no neighbor 192.168.0.1`	Removes the configuration for the exchange of information with the neighboring eBGP (ASBR) device. • The *ip-address* argument specifies the IP address of theneighbor.
Step 6	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits from address family configuration mode.
Step 7	**end** **Example:** `Device(config-router-af)# end`	Exits to privileged EXEC mode.

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.
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.

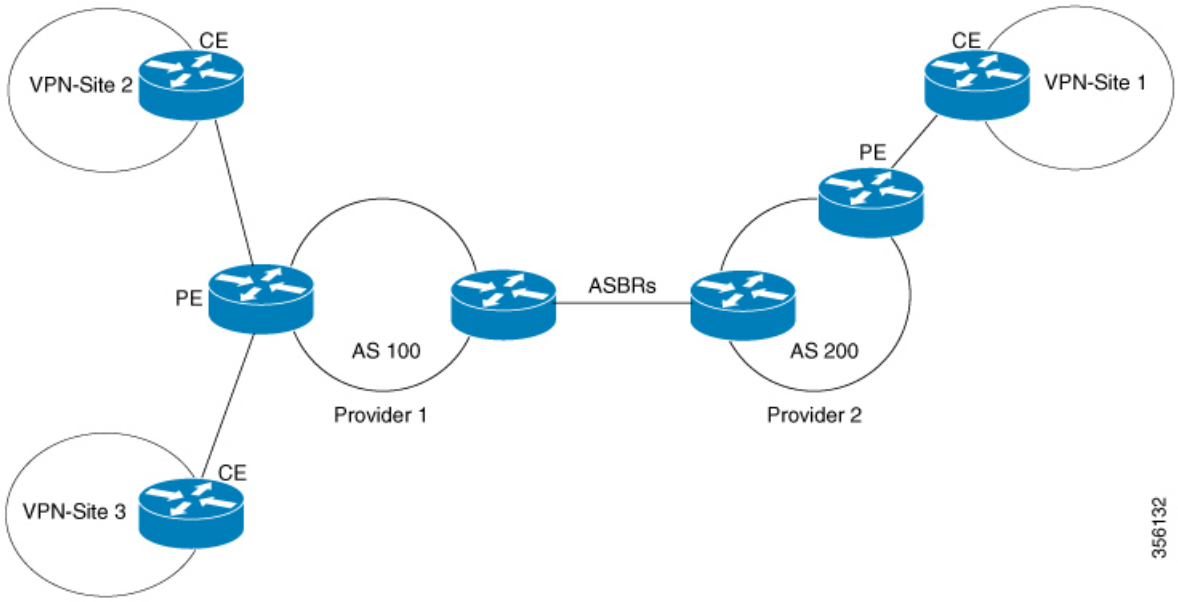
Command	Purpose
show { **ip** | **ipv6** } **route vrf** *vrf-name*	Displays the IP routing table that is associated with a VRF. Check that the loopback addresses of the local and remote CE routers are in the routing table of the PE routers.
show running-config bgp	Displays the running configuration 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

InterAS Option B

Next-Hop-Self Method

Figure 9: 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 2 exit-address-family		

Configuration for ASBR2 – P2 – PE2

Table 5:

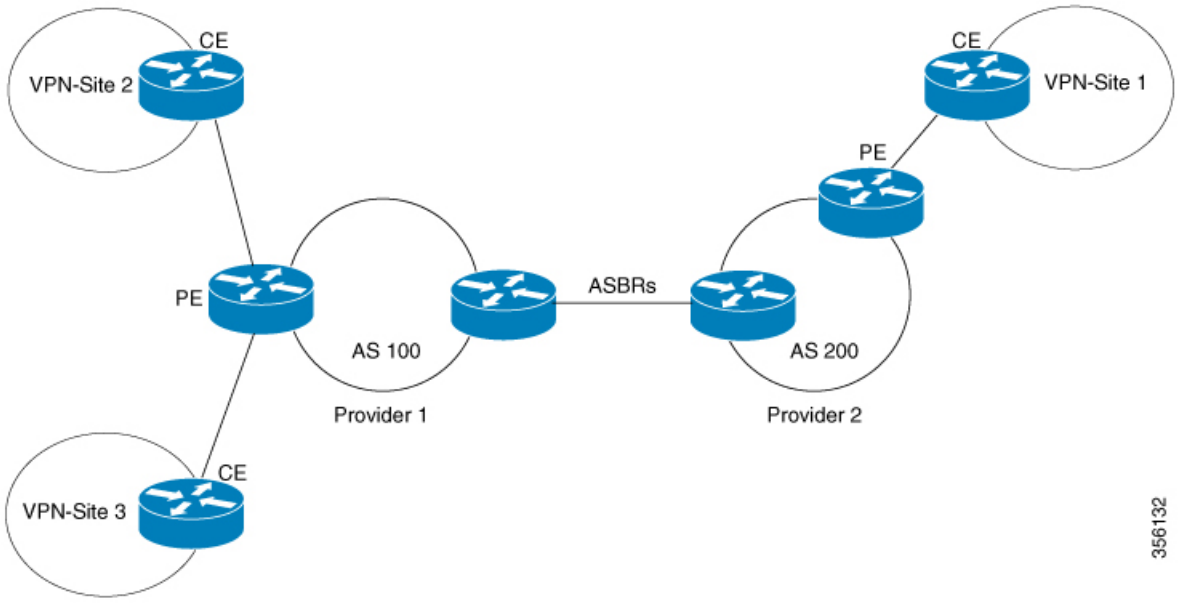
PE2	P2	ASBR2
	interface Loopback0 ip address 5.1.1.2 255.255.255.255 ip ospf 1 area 0 interface GigabitEthernet1/0/1 no switchport ip address 10.50.1.1 255.255.255.0 ip ospf 1 area 0 mpls ip mpls label protocol ldp interface GigabitEthernet2/0/3 no switchport ip address 10.40.1.2 255.255.255.0 ip ospf 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 2 exit-address-family		

IGP Redistribute Connected Subnets Method

Figure 10: 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 2 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 2 exit-address-family		

InterAS OptionAB

The following example displays the topology and the configuration on each device:

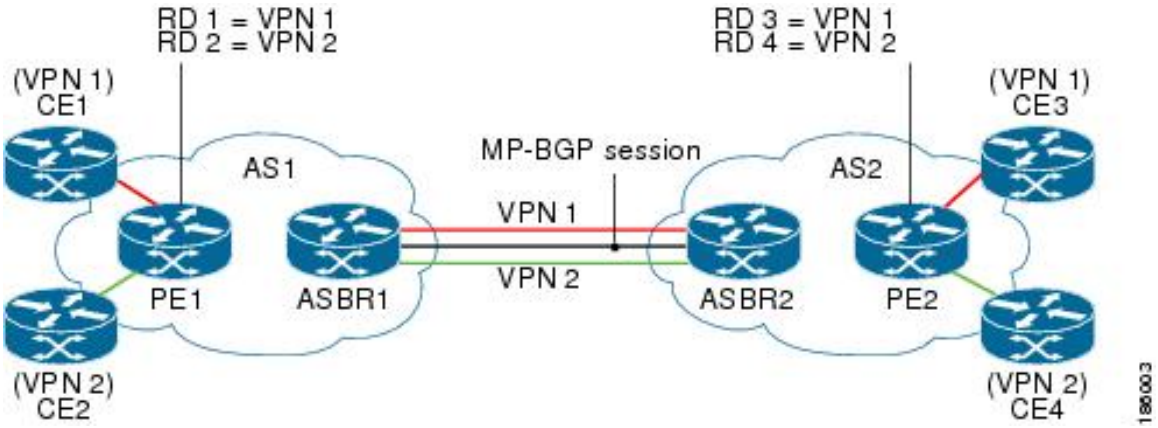


Table 6:

PE1 Config	P1 Config	ASBR1 Config	ASBR2 Config	PE2 Config
	interface Loopback0 ip address 2.2.2.2 255.255.255.255 ! interface TenGigabitEthernet1/1 ip address 10.1.1.2 255.255.255.0 mpls ip ! interface TenGigabitEthernet1/2 no ip address ! interface TenGigabitEthernet1/3 ip address 20.1.1.1 255.255.255.0 mpls ip ! router ospf 1 router-id 2.2.2.2 network 2.2.2.2 0.0.0.0 area 0 network 10.1.1.0 0.0.0.255 area 0 network 20.1.1.0 0.0.0.255 area 0 !			

PE1 Config	P1 Config	ASBR1 Config	ASBR2 Config	PE2 Config
ip vrf cust-1 rd 100:1 route-target export 100:1 route-target import 100:1 ! ip vrf cust-2 rd 100:2 route-target export 100:2 route-target import 100:2 ! interface Loopback0 ip address 1.1.1.1 255.255.255.255 ! interface Loopback1 ip address 11.11.11.11 255.255.255.255 ! interface Loopback2 ip address 12.12.12.12 255.255.255.255 ! ! interface HundredGigE1/0/1/1 no switchport ip address 10.1.1.1 255.255.255.0 mpls ip ! ! interface HundredGigE1/0/1/4 no switchport no ip address ! interface HundredGigE1/0/1/4.100 encapsulation dot1Q 100 ip vrf forwarding cust-1 ip address 11.1.1.1 255.255.255.0 ! interface HundredGigE1/0/1/4.101 encapsulation		ip vrf cust-1 rd 100:10001 route-target export 100:1 route-target import 100:1 route-target import 200:1 inter-as-hybrid next-hop 160.1.1.2 ! ip vrf cust-2 rd 100:20001 route-target export 100:2 route-target import 100:2 route-target import 200:2 inter-as-hybrid next-hop 170.1.1.2 ! ! interface Loopback0 ip address 3.3.3.3 255.255.255.255 ! ! interface TwentyFiveGigE1/0/3 no switchport ip address 20.1.1.2 255.255.255.0 mpls ip ! ! interface TwentyFiveGigE1/0/10.10 encapsulation dot1Q 10 ip address 150.1.1.1 255.255.255.0 mpls bgp forwarding ! interface TwentyFiveGigE1/0/10.20 encapsulation dot1Q 20 ip vrf forwarding cust-1 ip address 160.1.1.1 255.255.255.0 ! interface	ip vrf cust-1 rd 200:10001 route-target export 200:1 route-target import 200:1 route-target import 100:1 inter-as-hybrid next-hop 160.1.1.1 ! ip vrf cust-2 rd 200:20001 route-target export 200:2 route-target import 200:2 route-target import 100:2 inter-as-hybrid next-hop 170.1.1.1 ! ! interface Loopback0 ip address 4.4.4.4 255.255.255.255 ! ! interface TwentyFiveGigE1/0/2 no switchport ip address 30.1.1.1 255.255.255.0 mpls ip ! ! interface TwentyFiveGigE1/0/10.10 encapsulation dot1Q 10 ip address 150.1.1.2 255.255.255.0 mpls bgp forwarding ! interface TwentyFiveGigE1/0/10.20 encapsulation dot1Q 20 ip vrf forwarding cust-1 ip address 160.1.1.2 255.255.255.0	ip vrf cust-1 rd 200:1 route-target export 200:1 route-target import 200:1 route-target import 100:1 ! ip vrf cust-2 rd 200:2 route-target export 200:2 route-target import 200:2 route-target import 100:2 ! interface Loopback0 ip address 5.5.5.5 255.255.255.255 ! interface Loopback1 ip address 55.55.55.55 255.255.255.255 ! interface Loopback2 ip address 56.56.56.56 255.255.255.255 ! ! interface HundredGigE1/0/1/1.200 encapsulation dot1Q 200 ip vrf forwarding cust-1 ip address 55.1.1.1 255.255.255.0 ! interface HundredGigE1/0/1/1.201 encapsulation dot1Q 201 ip vrf forwarding cust-2 ip address 56.1.1.1 255.255.255.0 ! interface HundredGigE1/0/1/3 no switchport ip address

PE1 Config	P1 Config	ASBR1 Config	ASBR2 Config	PE2 Config
dot1Q 101 ip vrf forwarding cust-2 ip address 12.1.1.1 255.255.255.0 ! ! router ospf 2 vrf cust-1 router-id 11.11.11.11 network 11.1.1.0 0.0.0.255 area 0 network 11.11.11.11 0.0.0.0 area 0 ! router ospf 3 vrf cust-2 router-id 12.12.12.12 network 12.1.1.0 0.0.0.255 area 0 network 12.12.12.12 0.0.0.0 area 0 ! router ospf 1 router-id 1.1.1.1 network 1.1.1.1 0.0.0.0 area 0 network 10.1.1.0 0.0.0.255 area 0 ! router bgp 100 bgp router-id 1.1.1.1 bgp log-neighbor- changes neighbor 3.3.3.3 remote-as 100 neighbor 3.3.3.3 update- source Loopback0 ! address-family vpnv4 neighbor 3.3.3.3 activate neighbor 3.3.3.3 send- community extended exit-address-family ! address-family ipv4 vrf cust-1 redistribute connected		TwentyFiveGigE1/0/10.30 encapsulation dot1Q 30 ip vrf forwarding cust-2 ip address 170.1.1.1 255.255.255.0 ! router ospf 1 router-id 3.3.3.3 network 3.3.3.3 0.0.0.0 area 0 network 20.1.1.0 0.0.0.255 area 0 ! router bgp 100 bgp router-id 3.3.3.3 bgp log-neighbor- changes neighbor 1.1.1.1 remote- as 100 neighbor 150.1.1.2 remote-as 200 ! address-family ipv4 redistribute connected neighbor 1.1.1.1 activate neighbor 150.1.1.2 activate exit-address-family ! address-family vpnv4 neighbor 1.1.1.1 activate neighbor 1.1.1.1 send- community both neighbor 150.1.1.2 activate neighbor 150.1.1.2 send- community both neighbor 150.1.1.2 inter- as-hybrid exit-address-family ! address-family ipv4 vrf cust-1 redistribute connected exit-address-family	! interface TwentyFiveGigE1/0/10.30 encapsulation dot1Q 30 ip vrf forwarding cust-2 ip address 170.1.1.2 255.255.255.0 ! router ospf 1 router-id 4.4.4.4 network 4.4.4.4 0.0.0.0 area 0 network 30.1.1.0 0.0.0.255 area 0 ! router bgp 200 bgp router-id 4.4.4.4 bgp log-neighbor- changes neighbor 5.5.5.5 remote- as 200 neighbor 150.1.1.1 remote-as 100 ! address-family ipv4 neighbor 5.5.5.5 activate neighbor 150.1.1.1 activate exit-address-family ! address-family vpnv4 neighbor 5.5.5.5 activate neighbor 5.5.5.5 send-community both neighbor 150.1.1.1 activate neighbor 150.1.1.1 send-community both neighbor 150.1.1.1 inter-as-hybrid ' exit-address-family ! address-family ipv4 vrf cust-1 redistribute	30.1.1.2 255.255.255.0 mpls ip ! router ospf 2 vrf cust-1 router-id 55.55.55.55 network 55.1.1.0 0.0.0.255 area 0 network 55.55.55.55 0.0.0.0 area 0 ! router ospf 3 vrf cust-2 router-id 56.56.56.56 network 56.1.1.0 0.0.0.255 area 0 network 56.56.56.56 0.0.0.0 area 0 ! router ospf 1 router-id 5.5.5.5 network 5.5.5.5 0.0.0.0 area 0 network 30.1.1.0 0.0.0.255 area 0 ! router bgp 200 bgp router-id 5.5.5.5 bgp log-neighbor-changes neighbor 4.4.4.4 remote-as 200 neighbor 4.4.4.4 update-source Loopback0 ! address-family vpnv4 neighbor 4.4.4.4 activate neighbor 4.4.4.4 send-community extended exit-address-family ! address-family ipv4 vrf cust-1 redistribute connected redistribute ospf 2 maximum-paths ibgp

PE1 Config	P1 Config	ASBR1 Config	ASBR2 Config	PE2 Config
redistribute ospf 2 maximum-paths ibgp 2 exit-address-family ! address-family ipv4 vrf cust-2 redistribute connected redistribute ospf 3 maximum-paths ibgp 2 exit-address-family		! address-family ipv4 vrf cust-2 redistribute connected exit-address-family !	connected exit-address-family ! address-family ipv4 vrf cust-2 redistribute connected exit-address-family !	2 exit-address-family ! address-family ipv4 vrf cust-2 redistribute connected redistribute ospf 3 maximum-paths ibgp 2 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 9400 Series Switches)*

Feature History for MPLS VPN InterAS Options

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

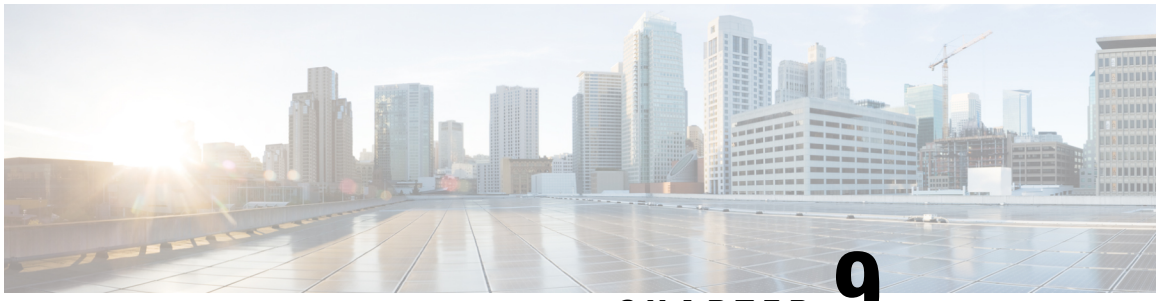
These features are available on all releases subsequent to the one they were introduced in, unless noted otherwise.

.

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.

Release	Feature	Feature Information
Cisco IOS XE Amsterdam 17.1.1	MPLS VPN InterAS Option A	MPLS VPN InterAS Option A is the simplest to configure of the available InterAS Options. This option provides back to back virtual routing and forwarding (VRF) connectivity. Here, MPLS VPN providers exchange routes across VRF interfaces.
Cisco IOS XE Amsterdam 17.3.1	MPLS VPN InterAS Option AB	MPLS VPN InterAS Option AB enables different autonomous systems to interconnect by using a single Multiprotocol Border Gateway Protocol (MP-BGP) session, which is enabled globally on the router.

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.

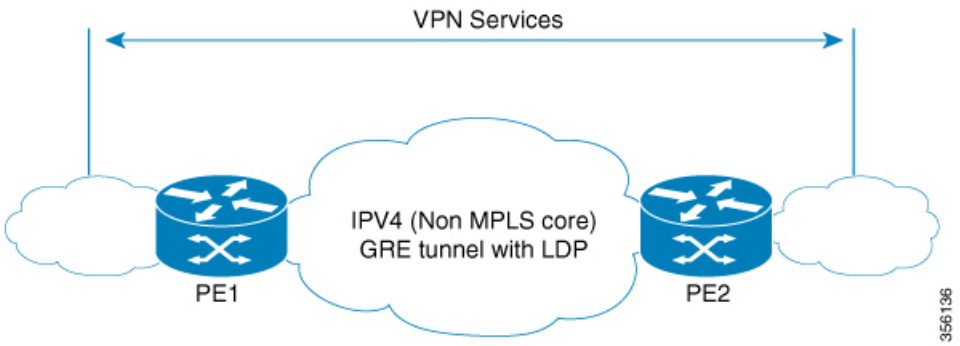
Note A similar nonscalable alternative is to connect each customer network through separate GRE tunnels (for example, connecting one customer network to each GRE tunnel).

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.

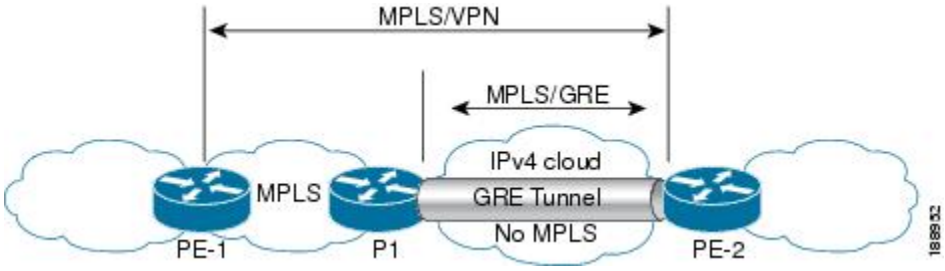
Figure 11: PE-to-PE Tunneling



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.

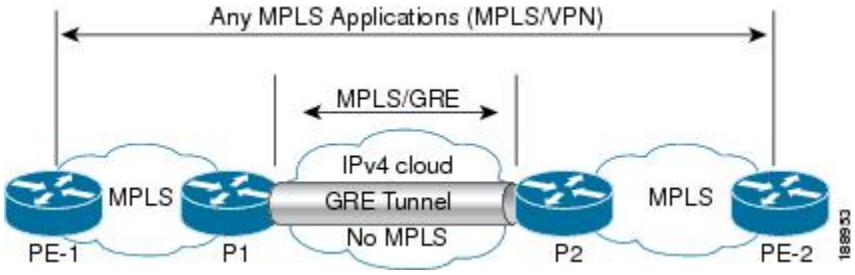
Figure 12: P-to-PE Tunneling



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 13: 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.

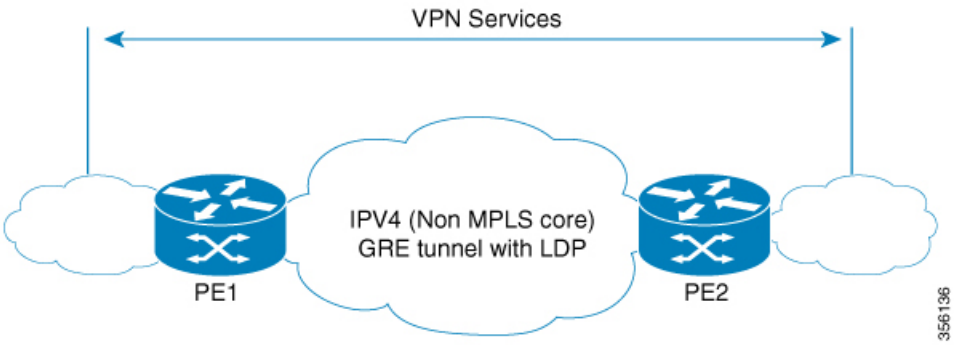
Configuration Examples for MPLS over GRE

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 14: 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

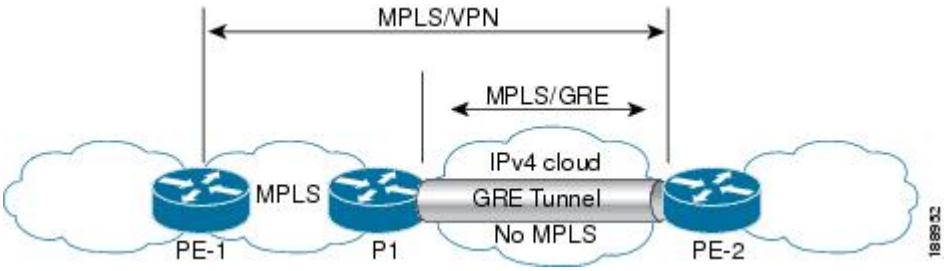
```
!
mpls ip
!
interface loopback 10
```

```
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 15: 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
!
```

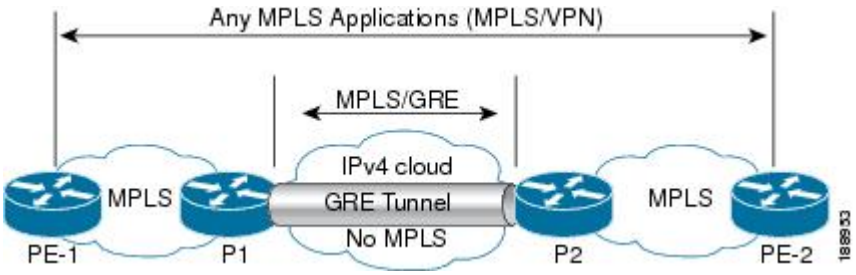
PE2 Configuration

```
!
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 16: 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 9400 Series Switches)*

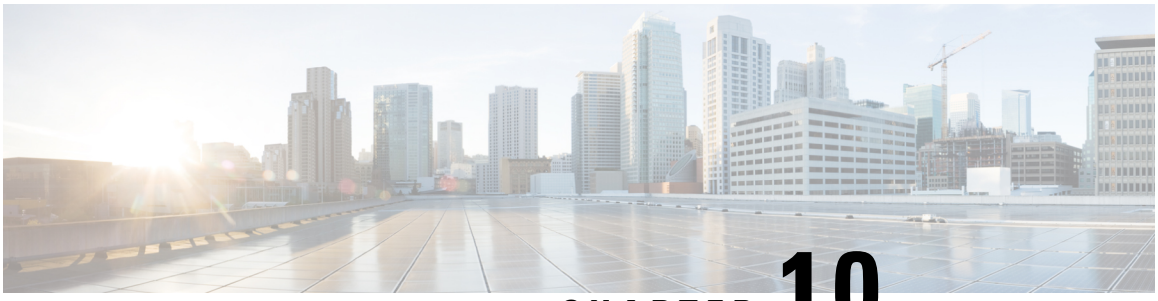
Feature History for MPLS over GRE

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.
Cisco IOS XE Cupertino 17.7.1	MPLS over GRE	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

Use the 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 10

Configuring MPLS Layer 2 VPN over GRE

Information About MPLS Layer 2 VPN over GRE

The MPLS Layer 2 VPN over GRE feature provides a mechanism for tunneling Multiprotocol Label Switching (MPLS) packets over non-MPLS networks. 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.

To configure MPLS Layer 2 VPN over GRE, you must have configured either Virtual Private LAN Service (VPLS) or EoMPLS (Ethernet over MPLS).

Types of Tunneling Configurations

The following sections provide information about the different types of tunneling configurations that are supported.

PE-to-PE Tunneling

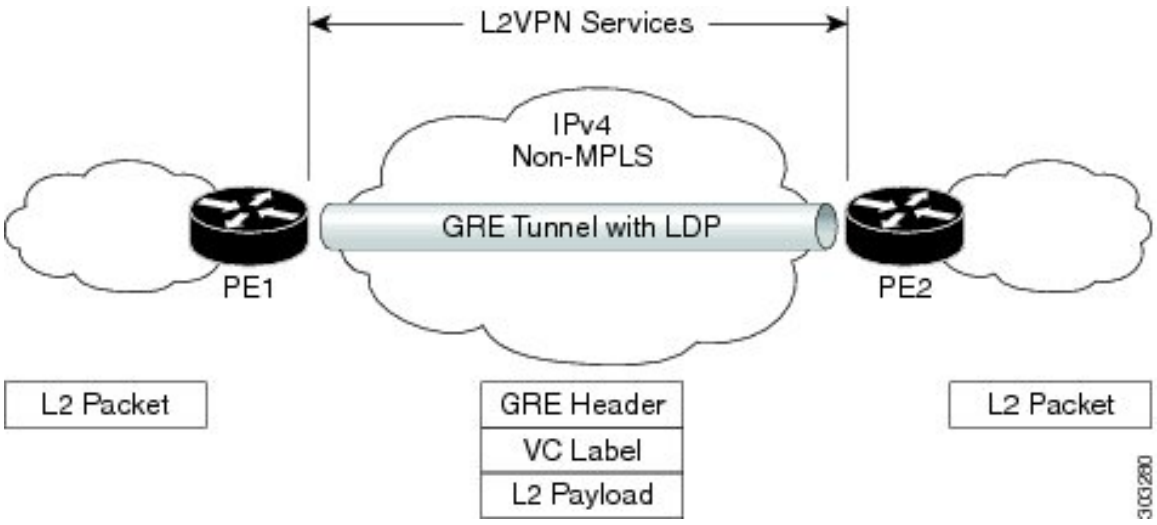
The provider edge-to-provider edge (PE-to-PE) tunneling configuration provides a scalable way to connect multiple customer networks across a non-MPLS network. With this configuration, traffic that is destined to multiple customer networks is multiplexed through a single GRE tunnel.

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 Border Gateway Protocol (BGP) 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.

Figure 17: PE-to-PE Tunneling, on page 162 shows an end-to-end IP core from one PE device to another through the GRE tunnel that spans the non-MPLS network.

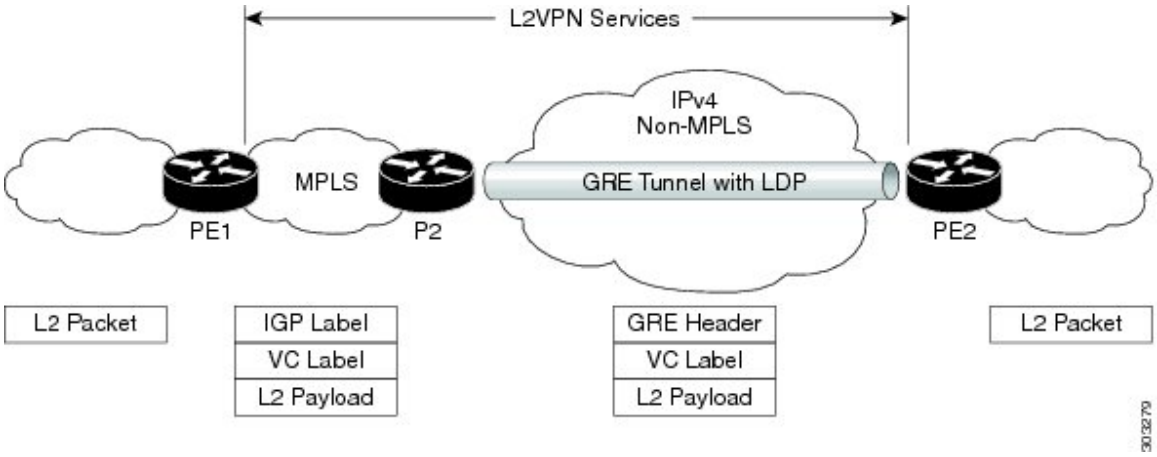
Figure 17: PE-to-PE Tunneling



P-to-PE Tunneling

Figure 18: P-to-PE Tunneling, on page 162 shows a method of connecting two MPLS segments (P2 to PE2) 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 GRE tunnel.

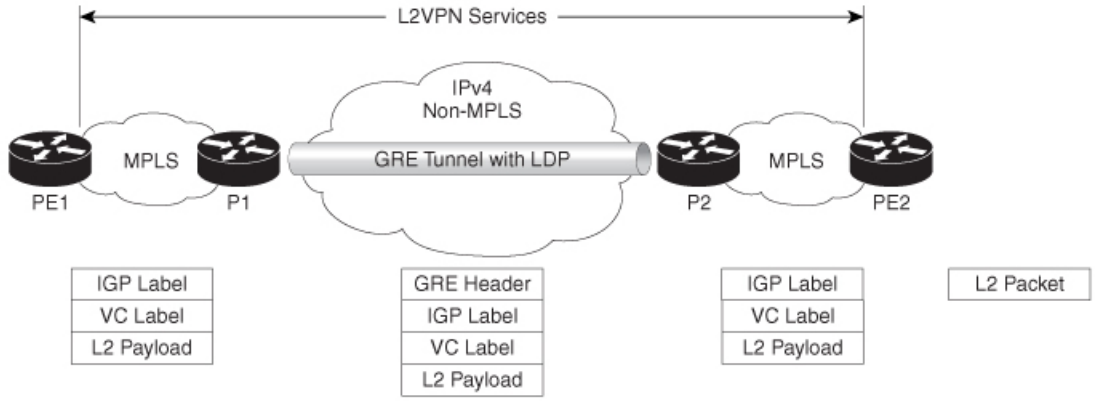
Figure 18: P-to-PE Tunneling



P-to-P Tunneling

Figure 19: P-to-P Tunneling, on page 163 shows a method of connecting two 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 GRE tunnel.

Figure 19: P-to-P Tunneling



How to Configure MPLS Layer 3 VPN over GRE

To configure the MPLS over GRE feature, you must create a GRE tunnel to span the non-MPLS networks. Perform the following procedure on the devices that are 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`	Configures the tunnel's source IP address.

	Command or Action	Purpose
Step 6	**tunnel destination** *destination-address* **Example:** `Device(config-if)# tunnel destination 10.1.1.2`	Configures the tunnel's destination IP address.
Step 7	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS on the tunnel's physical interface.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuration Examples for MPLS Layer 2 VPN over GRE

The following section provides an example for configuring MPLS Layer 2 VPN over GRE.

Example: Configuring a GRE Tunnel That Spans a non-MPLS Network

The following examples show how to configure a generic GRE tunnel configuration that spans a non-MPLS network.

The following example shows the tunnel configuration on the PE1 device:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel 1
Device(config-if)# ip address 10.1.1.1 255.255.255.0
Device(config-if)# tunnel source 10.0.0.1
Device(config-if)# tunnel destination 10.0.0.2
Device(config-if)# ip ospf 1 area 0
Device(config-if)# mpls ip
```

The following example shows the tunnel configuration on the PE2 device:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel 1
Device(config-if)# ip address 10.1.1.2 255.255.255.0
Device(config-if)# tunnel source 10.0.0.2
Device(config-if)# tunnel destination 10.0.0.1
Device(config-if)# ip ospf 1 area 0
Device(config-if)# mpls ip
```

Additional References for Configuring MPLS Layer 2 VPN over GRE

Related Documents

Related Topic	Document Title
Configuring VPLS	For more information, see Information About VPLS.
Configuring Ethernet-over-MPLS (EoMPLS) and Pseudowire Redundancy (PWR)	For more information, see How to Configure Ethernet Over MPLS.

Feature History for Configuring MPLS Layer 2 VPN over GRE

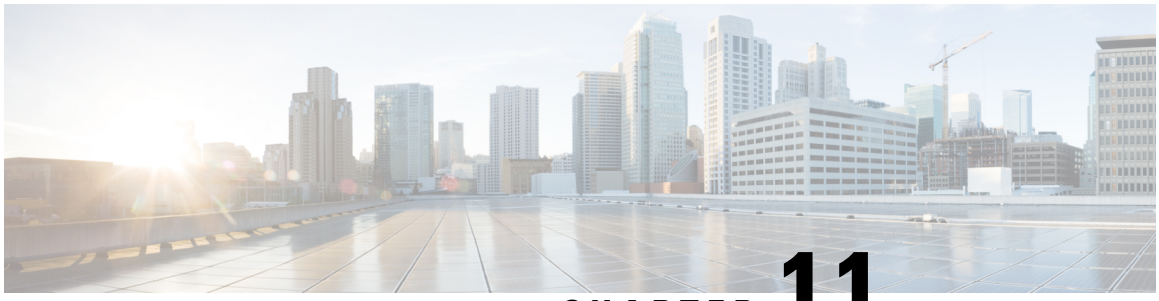
This table provides release and related information for the features explained in this module.

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 16.12.1	MPLS Layer 2 VPN over GRE	The MPLS Layer 2 VPN over GRE feature provides a mechanism for tunneling MPLS packets over non-MPLS networks.
Cisco IOS XE Cupertino 17.7.1	MPLS Layer 2 VPN over GRE	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 11

Configuring MPLS Layer 3 VPN over GRE

Prerequisites for MPLS Layer 3 VPN over GRE

- Ensure that your Multiprotocol Label Switching (MPLS) virtual private network (VPN) is configured.
- Ensure that the following routing protocols are configured:
  - Label Distribution Protocol (LDP): For MPLS label distribution.
  - Multiprotocol Border Gateway Protocol (MP-BGP): For VPN route and label distribution.
- We recommend that you use the Quality of Service (QoS) group value for defining QoS policies to traverse the protocol boundary. QoS group values are required because the ingress traffic enters the IP core from the MPLS network and the egress traffic leaves the IP core to enter the MPLS network.
- Before configuring a generic routing encapsulation (GRE) tunnel, configure a loopback interface (that is not attached to a virtual routing and forwarding [VRF]) interface with an IP address. This dummy loopback interface with an IPv4 address enables the internally created tunnel interface for IPv4 forwarding. You do not have to configure a loopback interface if the system has at least one interface that is not attached to the VRF and is configured with an IPv4 address.

Restrictions for MPLS Layer 3 VPN over GRE

The MPLS Layer 3 VPN over GRE feature does not support the following:

- QoS service policies that are configured on the tunnel interface

Note Although QoS service policies configured on the tunnel interface are not supported, QoS service policies configured on a physical interface or a sub-interface are supported.

- GRE options such as sequencing, checksum, and source route
- IPv6 GRE configurations
- Advanced features such as Carrier Supporting Carrier (CSC)

Information About MPLS Layer 3 VPN over GRE

The MPLS Layer 3 VPN over GRE feature provides a mechanism for tunneling MPLS packets over non-MPLS networks. This feature allows you to create a 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.

Types of Tunneling Configurations

The following sections provide information about the different types of tunneling configurations that are supported.

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 GRE tunnel.

As shown in the Figure 20: PE-to-PE Tunneling, on page 169, the PE devices assign VRF numbers to the customer edge (CE) devices on each side of the non-MPLS network.

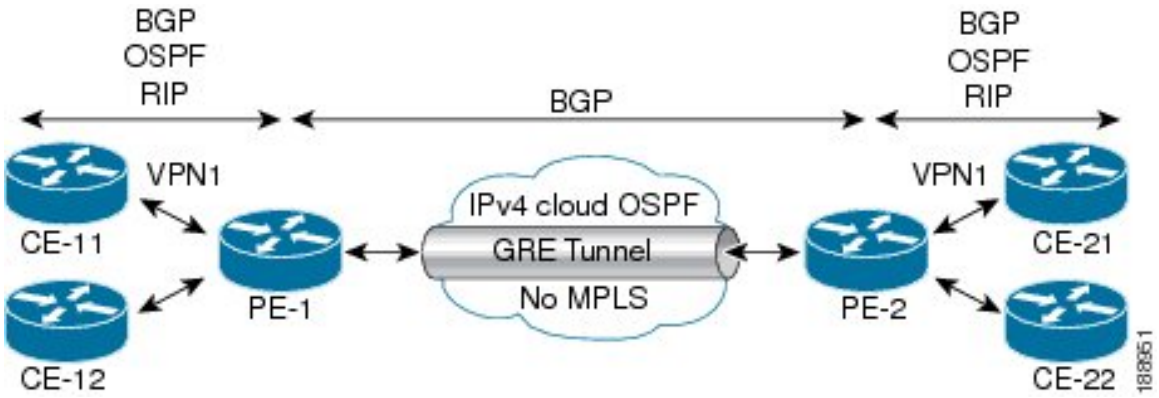
The PE devices use routing protocols such as Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), or Routing Information Protocol (RIP) to learn about the IP networks behind the CE devices. The routes to the IP networks behind the CE devices are stored in the associated CE device's VRF routing table.

The PE device on one side of the non-MPLS network uses 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 BGP 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.

Figure 20: PE-to-PE Tunneling, on page 169 shows BGP defining a static route to the BGP neighbor (the opposing PE device) through the GRE tunnel that spans the non-MPLS network. Because the routes that are learned by the BGP neighbor include the GRE tunnel next hop, all the customer network traffic is sent using the GRE tunnel.

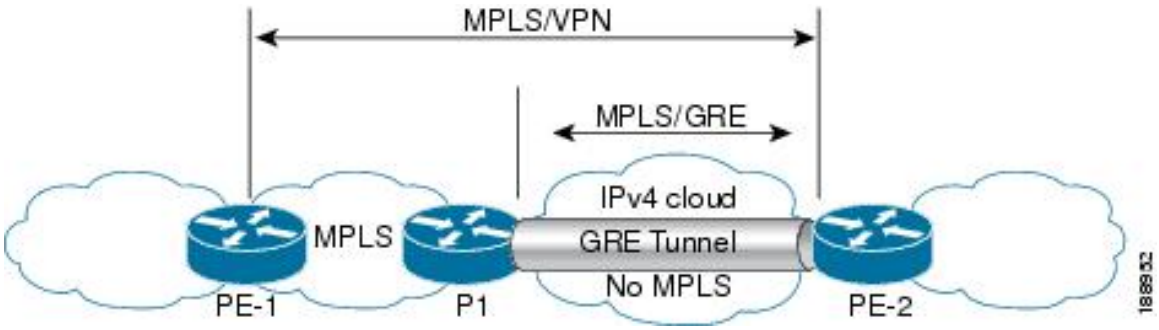
Figure 20: PE-to-PE Tunneling



P-to-PE Tunneling

Figure 21: P-to-PE Tunneling, on page 169 shows a method of connecting two MPLS segments (P2 to PE2) 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 GRE tunnel.

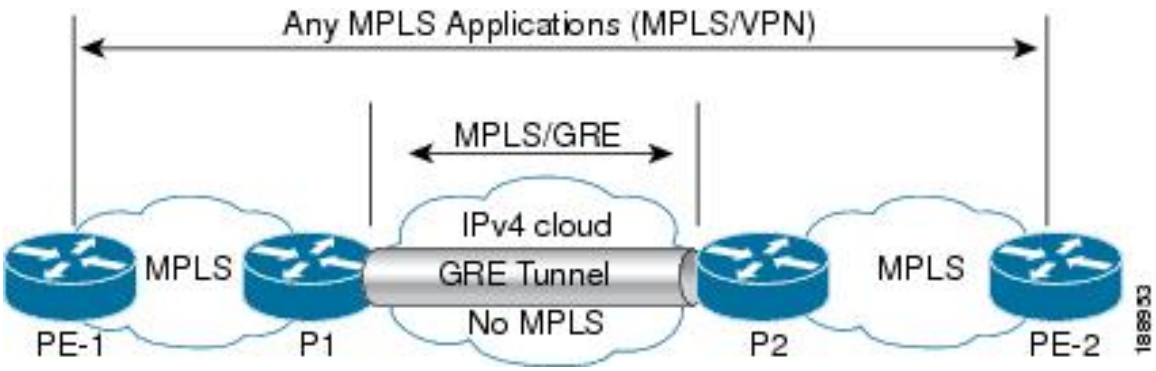
Figure 21: P-to-PE Tunneling



P-to-P Tunneling

Figure 22: P-to-P Tunneling, on page 170 shows a method of connecting two 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 GRE tunnel.

Figure 22: P-to-P Tunneling



How to Configure MPLS Layer 3 VPN over GRE

To configure the MPLS over GRE feature, you must create a GRE tunnel to span the non-MPLS networks. Perform the following procedure on the devices that are 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`	Configures the tunnel's source IP address.
Step 6	**tunnel destination** *destination-address* **Example:** `Device(config-if)# tunnel destination 10.1.1.2`	Configures the tunnel's destination IP address.

	Command or Action	Purpose
Step 7	**mpls ip** **Example:** `Device(config-if)# mpls ip`	Enables MPLS on the tunnel's physical interface.
Step 8	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuration Examples for MPLS Layer 3 VPN over GRE

The following sections provide various configuration examples for MPLS Layer 3 VPN over GRE.

Example: Configuring MPLS Layer 3 VPN over GRE (PE-to-PE Tunneling)

The following examples show how to configure Layer 3 VPN and the GRE tunnel from PE1 to PE2 (see Figure 20: PE-to-PE Tunneling, on page 169).

The following example shows how to configure a loopback interface on PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface Loopback10
Device(config-if)# ip address 209.165.200.225 255.255.255.255
Device(config-if)# end
```

The following example shows how to configure a loopback interface on PE2:

```
Device> enable
Device# configure terminal
Device(config)# interface Loopback3
Device(config-if)# ip address 209.165.202.129 255.255.255.255
Device(config-if)# end
```

The following example shows how to advertise a loopback in IGP on PE1:

```
Device> enable
Device# configure terminal
Device(config)# router ospf 10
Device(config-router)# router-id 198.51.100.10
Device(config-router)# end
```

The following example shows how to configure a GRE tunnel, configure a different IGP instance on the tunnel, and enable MPLS on the tunnel on PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel13
Device(config-if)# ip address 203.0.113.200 255.255.255.248
Device(config-if)# ip ospf 11 area 0
Device(config-if)# mpls ip
Device(config-if)# tunnel source 209.165.200.225
Device(config-if)# tunnel destination 209.165.202.129
Device(config-if)# end
```

The following example shows how to configure a GRE tunnel, configure a different IGP instance on the tunnel, and enable MPLS on the tunnel on PE2:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel31
Device(config-if)# ip address 203.0.113.201 255.255.255.248
Device(config-if)# ip ospf 11 area 0
Device(config-if)# mpls ip
Device(config-if)# tunnel source 209.165.202.129
Device(config-if)# tunnel destination 209.165.200.225
Device(config-if)# end
```

The following example shows how to advertise PE1 loopback IP for BGP in IGP instance configured on the tunnel:

```
Device> enable
Device# configure terminal
Device(config)# router ospf 11
Device(config-router)# router-id 198.51.100.11
Device(config-router)# network 192.0.1.1 0.0.0.0 area 0
Device(config-router)# end
```

The following example shows how to advertise PE2 loopback IP for BGP in IGP instance configured on the tunnel:

```
Device> enable
Device# configure terminal
Device(config)# router ospf 11
Device(config-router)# router-id 203.0.113.201
Device(config-router)# network 192.0.1.1 0.0.0.0 area 0
Device(config-router)# end
```

The following example shows how to configure VRF on PE1 where CE1 is connected:

```
Device> enable
Device# configure terminal
Device(config)# vrf definition vrf-1
Device (config-vrf)# rd 1:1
Device (config-vrf)# address-family ipv4
Device (config-vrf-af)# route-target import 1:2
Device (config-vrf-af)# route-target export 1:1
Device(config-vrf)# end
```

The following example shows how to configure VRF on PE2 where CE2 is connected:

```
Device> enable
Device# configure terminal
Device (config)# vrf definition vrf-1
Device (config-vrf)# rd 2:2
Device (config-vrf)# address-family ipv4
Device (config-vrf-af)# route-target import 1:1
Device (config-vrf-af)# route-target export 1:2
Device(config-vrf)# end
```

The following example shows how to configure PE1-CE1 interface:

```
Device> enable
Device# configure terminal
Device (config)# int po14.1
Device (config-subif)# encapsulation dot1Q 10
Device (config-subif)# vrf forwarding vrf-1
Device (config-subif)# ip address 14.2.1.1 255.255.255.0
Device(config-subif)# end
```

The following example shows how to configure PE2-CE2 interface:

```
Device> enable
Device# configure terminal
Device (config)# int po24.1
Device (config-subif)# encapsulation dot1Q 10
Device (config-subif)# vrf forwarding vrf-1
Device (config-subif)# ip address 24.2.1.1 255.255.255.0
Device(config-subif)# end
```

The following example shows how to configure PE1-CE1 External Border Gateway Protocol (EBGP):

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# address-family ipv4 vrf vrf-1
Device (config-router-af)# neighbor 14.2.1.2 remote-as 65041
Device (config-router-af)# neighbor 14.2.1.2 activate
Device (config-router-af)# exit-address-family
Device(config-router)# end
```

The following example shows how to configure PE2-CE2 EBGP:

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# address-family ipv4 vrf vrf-1
Device (config-router-af)# neighbor 24.2.1.2 remote-as 65041
Device (config-router-af)# neighbor 24.2.1.2 activate
Device (config-router-af)# exit-address-family
Device (config-router)# end
```

The following example shows how to configure PE1-PE2 MP-BGP on PE1:

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# neighbor 192.0.2.1 remote-as 65040
Device (config-router)# neighbor 192.0.2.1 update-source Loopback0
Device (config-router)# address-family ipv4
Device (config-router-af)# neighbor 192.0.2.1 activate
Device (config-router-af)# exit
Device (config-router)# address-family vpnv4
Device (config-router-af)#  neighbor 192.0.2.1 activate
Device (config-router-af)#  neighbor 192.0.2.1 send-community both
Device (config-router-af)# exit
Device (config-router)# end
```

Example: Configuring MPLS Layer 3 VPN over GRE (P-to-PE Tunneling)

The following examples show how to configure Layer 3 VPN on the PE devices (PE1 and PE2) and MPLS segment (P1), and the GRE tunnel from PE1 to P1 to PE2 (see Figure 21: P-to-PE Tunneling, on page 169).

The following example shows how to configure loopback interface for GRE tunnel for PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface Loopback4
```

```
Device(config-if)# ip address 209.165.200.230 255.255.255.255
Device(config-if)# end
```

The following example shows how to configure loopback interface for GRE tunnel for P1:

```
Device> enable
Device# configure terminal
Device(config)# interface Loopback100
Device(config-if)# ip address 209.165.200.235 255.255.255.255
Device(config-if)# end
```

The following example shows how to configure interface from PE1-P1 and configure IGP:

```
Device> enable
Device# configure terminal
Device(config)# interface Port-channel11
Device(config-if)# no switchport
Device(config-if)# ip address 209.165.201.1 255.255.255.248
Device(config-if)# ip ospf 10 area 0
Device(config-if)# end
```

The following example shows how to configure interface from P1-PE1 and configure IGP:

```
Device> enable
Device# configure terminal
Device(config)# interface Port-channel1
Device(config-if)# no switchport
Device(config-if)# ip address 209.165.201.2 255.255.255.248
Device(config-if)# ip broadcast-address 209.165.201.31
Device(config-if)# ip ospf 10 area 0
Device(config-if)# end
```

The following example shows how to advertise loopback in IGP on PE1:

```
Device> enable
Device# configure terminal
Device(config)# router ospf 10
Device(config-router)# router-id 198.51.100.10
Device(config-router)# network 209.165.200.230 0.0.0.0 area 0
Device(config-router)# end
```

The following example shows how to advertise loopback in IGP on P1:

```
Device> enable
Device# configure terminal
Device(config)# router ospf 10
Device(config-router)# router-id 198.51.100.20
Device(config-router)# network 209.165.200.235 0.0.0.0 area 0
Device(config-router)# end
```

The following example shows how to configure GRE tunnel, configure an IGP instance on the tunnel, and enable MPLS on the tunnel on PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel111
Device(config-if)# ip address 209.165.202.140 255.255.255.248
Device(config-if)# ip ospf 11 area 0
Device(config-if)# mpls ip
Device(config-if)# tunnel source 209.165.200.230
Device(config-if)# tunnel destination 209.165.200.235
Device(config-if)# end
```

The following example shows how to configure GRE tunnel, configure an IGP instance on the tunnel, and enable MPLS on the tunnel on P1:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel111
Device(config-if)# ip address 209.165.202.141 255.255.255.248
Device(config-if)# ip ospf 11 area 0
Device(config-if)# mpls ip
Device(config-if)# tunnel source 209.165.200.235
Device(config-if)# tunnel destination 209.165.200.230
Device(config-if)# end
```

The following example shows how to advertise PE loopback IP for BGP in tunnel's IGP instance on PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface Tunnel111
Device(config)# router ospf 11
Device(config-router)# router-id 198.51.100.11
Device(config-router)# network 192.0.1.1 0.0.0.0 area 0
Device(config-router)# end
```

The following example shows how to configure interface from PE2-P1, and configure IGP and MPLS:

```
Device> enable
Device# configure terminal
Device(config)# interface Port-channel12
Device(config-if)# no switchport
Device(config-if)# ip address 209.165.201.1 255.255.255.248
Device(config-if)# ip ospf 11 area 0
Device(config-if)# mpls ip
Device(config-if)# end
```

The following example shows how to configure interface from P1-PE2, and configure IGP:

```
Device> enable
Device# configure terminal
Device(config)# interface Port-channel12
Device(config-if)# no switchport
Device(config-if)# ip address 209.165.201.2 255.255.255.248
Device(config-if)# ip ospf 11 area 0
Device(config-if)# mpls ip
Device(config-if)# end
```

The following example shows how to create VRF on PE1 where CE1 is connected:

```
Device> enable
Device# configure terminal
Device(config)# vrf definition vrf-1
Device (config-vrf)# rd 1:1
Device (config-vrf)# address-family ipv4
Device (config-vrf-af)# route-target import 1:2
Device (config-vrf-af)# route-target export 1:1
Device (config-vrf-af)# exit
Device (config-vrf)# end
```

The following example shows how to create VRF on PE2 where CE2 is connected:

```
Device> enable
Device# configure terminal
Device (config)# vrf definition vrf-1
Device (config-vrf)# rd 2:2
Device (config-vrf)# address-family ipv4
Device (config-vrf-af)# route-target import 1:1
Device (config-vrf-af)# route-target export 1:2
```

```
Device (config-vrf-af)# exit
Device (config-vrf)# end
```

The following example shows how to configure PE1-CE1 interface:

```
Device> enable
Device# configure terminal
Device (config)# int po14.1
Device (config-subif)# encapsulation dot1Q 10
Device (config-subif)# vrf forwarding vrf-1
Device (config-subif)# ip address 14.2.1.1 255.255.255.0
Device (config-subif)# exit
Device (config)# end
```

The following example shows how to configure PE2-CE2 interface:

```
Device> enable
Device# configure terminal
Device (config)# int po24.1
Device (config-subif)# encapsulation dot1Q 10
Device (config-subif)# vrf forwarding vrf-1
Device (config-subif)# ip address 24.2.1.1 255.255.255.0
Device (config-subif)# exit
Device (config)# end
```

The following example shows how to configure PE1-CE1 EBGP:

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# address-family ipv4 vrf vrf-1
Device (config-router-af)# neighbor 14.2.1.2 remote-as 65041
Device (config-router-af)# neighbor 14.2.1.2 activate
Device (config-router-af)# exit-address-family
Device (config-router)# end
```

The following example shows how to configure PE2-CE2 EBGP:

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# address-family ipv4 vrf vrf-1
Device (config-router-af)# neighbor 24.2.1.2 remote-as 65041
Device (config-router-af)# neighbor 24.2.1.2 activate
Device (config-router-af)# exit-address-family
Device (config-router)# end
```

The following example shows how to configure PE1-PE2 MP-BGP on PE1:

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# neighbor 192.0.2.1 remote-as 65040
Device (config-router)# neighbor 192.0.2.1 update-source Loopback0
Device (config-router)# address-family ipv4
Device (config-router-af)# neighbor 192.0.2.1 activate
Device (config-router-af)# exit
Device (config-router)# address-family vpnv4
Device (config-router-af)# neighbor 192.0.2.1 activate
Device (config-router-af)# neighbor 192.0.2.1 send-community both
Device (config-router-af)# exit
Device (config-router)# end
```

The following example shows how to configure PE2-PE1 MP-BGP on PE2:

```
Device> enable
Device# configure terminal
Device (config)# router bgp 65040
Device (config-router)# neighbor 192.0.1.1 remote-as 65040
Device (config-router)# neighbor 192.0.1.1 update-source Loopback0
Device (config-router)# address-family ipv4
Device (config-router-af)# neighbor 192.0.1.1 activate
Device (config-router-af)# exit
Device (config-router)# address-family vpnv4
Device (config-router-af)# neighbor 192.0.1.1 activate
Device (config-router-af)# neighbor 192.0.1.1 send-community both
Device (config-router-af)# exit
Device (config-router)# end
```

Feature History for Configuring MPLS Layer 3 VPN over GRE

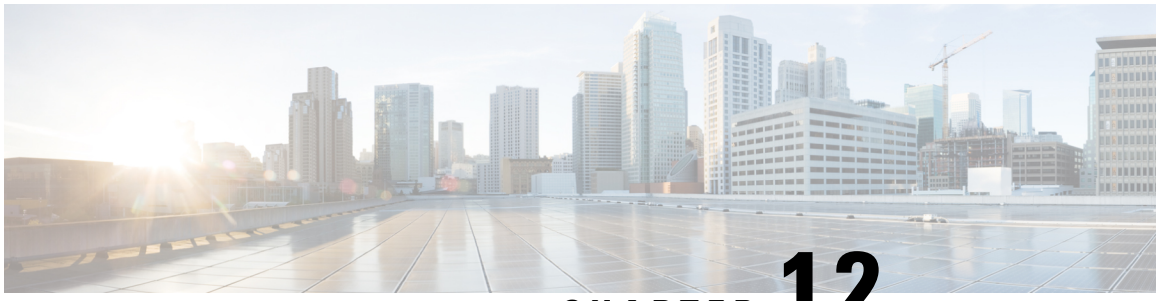
This table provides release and related information for the features explained in this module.

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 16.12.1	MPLS Layer 3 VPN over GRE	The MPLS Layer 3 VPN over GRE feature provides a mechanism for tunneling MPLS packets over a non-MPLS network.
Cisco IOS XE Cupertino 17.7.1	MPLS Layer 3 VPN over GRE	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 12

Configuring MPLS QoS

Prerequisites for MPLS QoS

- The switch must be configured as a Multiprotocol Label Switching (MPLS) provider edge (PE) or provider (P) router, which includes 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 MPLS QoS

This section provides detailed information about MPLS QoS.

MPLS QoS Overview

The MPLS QoS functionality enables network administrators to provide differentiated services across an MPLS network. Network administrators can satisfy a wide range of networking requirements by specifying the CoS applicable to each transmitted IP packet. Different classes of service can be established for IP packets by setting the IP precedence bit in the header of each packet. Classification, remarking, and queuing on an MPLS network is performed over MPLS EXP bits. In the MPLS network, the packets are differentiated by the MPLS EXP field marking, and are treated accordingly, depending on the weighted early random detection (WRED) configuration.

The MPLS EXP field in MPLS packet 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 “Classifying Network Traffic”.

- 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 “Marking Network Traffic”.

- Queueing

  Queueing helps prevent traffic congestion. This includes priority level queueing, weighted tail drop (WTD), scheduling, shaping and weighted random early detection (WRED) features.

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.

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.

WRED monitors network traffic to anticipate and prevent congestion at common network and internetwork bottlenecks. WRED can selectively discard lower priority traffic when an interface becomes congested. This feature can also provide differentiated performance characteristics for different classes of service.

There are two ways to transport packets through the MPLS network:

Uniform mode: Uniform mode of transferring packets operates on one layer of QoS. The Provider Edge at ingress copies the DSCP information from the incoming IP packet into the MPLS EXP bits of the imposed labels and the IP precedence bits are mapped to the MPLS EXP field . As the EXP bits travel through the core, they may or may not be modified by the intermediate devices on the network. The Provider Edge at egress copies the EXP bits to the DSCP bits of the newly exposed IP packet.

Pipe mode: Pipe mode of transferring packets operates on two layers of QoS. An underlying QoS for the data that remains unchanged when traversing the core. A per-core QoS, which is separate from that of the underlying IP packets. The DSCP information is saved and stored as the packet travels through the MPLS network. The MPLS EXP label is applied by the PE at ingress but the IP precedence bits are not stored. At egress, the original IP precedence value is preserved.

Benefits of MPLS EXP Classification and Marking

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. 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 Configure MPLS QoS

This section provides information about how to configure MPLS QoS:

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.

	Command or Action	Purpose
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

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

Marking MPLS EXP on Label Switched Packets

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.

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

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.
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.
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 exp table** *table-map-name* **Example:** `Device(config-pmap-c-police)# exceed-action set-mpls-exp-topmost-transmit exp table dscp2exp`	Defines the action to take on packets that exceed the values specified by the policer.

	Command or Action	Purpose
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.

Configuring WRED for MPLS EXP

Perform this task to enable WRED for MPLS EXP.

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 wred_exp`	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 exp`	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	**bandwidth** {*kbps* | **remaining***percentage* | **percent***percentage*} **Example:**	Specify either the bandwidth allocated for a class belonging to a policy map or the traffic shaping.

	Command or Action	Purpose
	`Device(config-pmap-c)# bandwidth percent 30`	
Step 6	**random-detect mpls-exp-based** **Example:** `Device(config-pmap-c)# random-detect mpls-exp- based`	Configures WRED to use the MPLS EXP value when it calculates the drop probability for the packet.

Step 7	**random-detect exp** *exp-value* **percent** *min-threshold max-threshold*
	Example:
	`Device(config-pmap-c)# random-detect exp 1 10 20`
	`Device(config-pmap-c)# random-detect exp 2 30 40`
	`Device(config-pmap-c)# random-detect exp 2 40 80`
		Specifies the MPLS EXP value, minimum and maximum thresholds, in percentage.

Step 8	**end** **Example:** `Device(config-pmap-c-police)# end`	(Optional) Returns to privileged EXEC mode.

Configuration Examples for MPLS QoS

This section provides configuration examples for MPLS QoS.

Example: Classifying MPLS Encapsulated Packets

Defining an MPLS EXP Class Map

The following example shows how to define 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 shows how to use the class map created in the example above to define a policy map. This example also shows how to applys 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 shows how to define a policy map that sets the MPLS EXP top-most 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 on 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 following example shows how to create 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
```

Example: Configuring WRED for MPLS EXP

The following example shows how to enable WRED for MPLS EXP:

```
Device# configure terminal
Device(config)# policy-map wred_exp
Device(config-pmap-c)# bandwidth percent 30
```

```
Device(config-pmap-c)# random-detect mpls-exp-based
Device(config-pmap-c)# random-detect exp 1 10 20
Device(config-pmap-c)# random-detect exp 2 30 40
Device(config-pmap-c)# random-detect exp 2 40 80
```

Displaying WRED Threshold Labels

The **show policy-map** *policy-map-name* command verifies the WRED configuration for MPLS EXP.

The following sample output displays WRED threshold labels:

```
Device# show policy-map wred_exp
Policy Map wred_exp
Class exp
bandwidth 30 (%)
percent-based wred, exponential weight 9
exp    min-threshold    max-threshold
---------------------------------------
 0             -                -
 1             10               20
 2             30               40
 3             40               80
 4             -                -
 5             -                -
 6             -                -
 7             -                -
```

Additional References

Related Documents

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

Feature History for QoS MPLS EXP

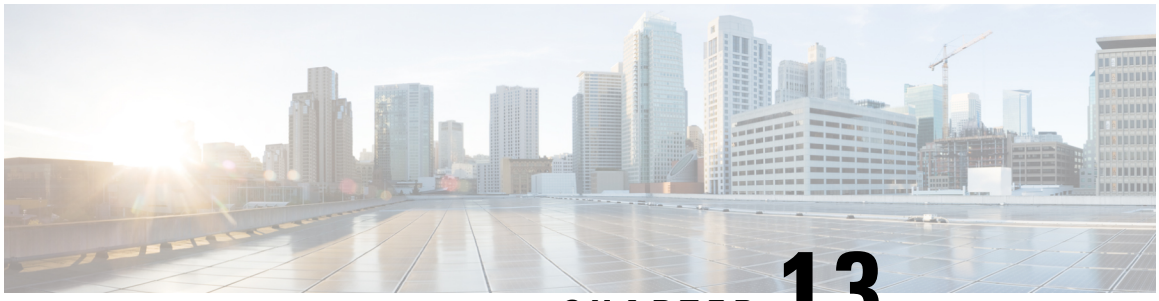
This table provides release and related information for features explained in this module.

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 16.6.1	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.

Release	Feature	Feature Information
Cisco IOS XE Amsterdam 17.3.1	MPLS QoS - WRED	Introduces support for weighted random early detection (WRED) in MPLS Quality of Service (QoS). This feature configures WRED to use the MPLS experimental bits (EXP) to calculate the drop probability of a packet.
Cisco IOS XE Cupertino 17.7.1	QoS MPLS EXP	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 13

Configuring MPLS Static Labels

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

- 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 is not supported.
- MPLS Static Labels is not supported for label-controlled Asynchronous Transfer Mode (lc-atm).
- MPLS static bindings are not supported for local prefixes.
- VRF aware Static Labels is not supported,

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 the binding between a label and an IPv4 prefix statically.

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.

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.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
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: 16/983039
   [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.

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 16–983039 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: 16/983039
   [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

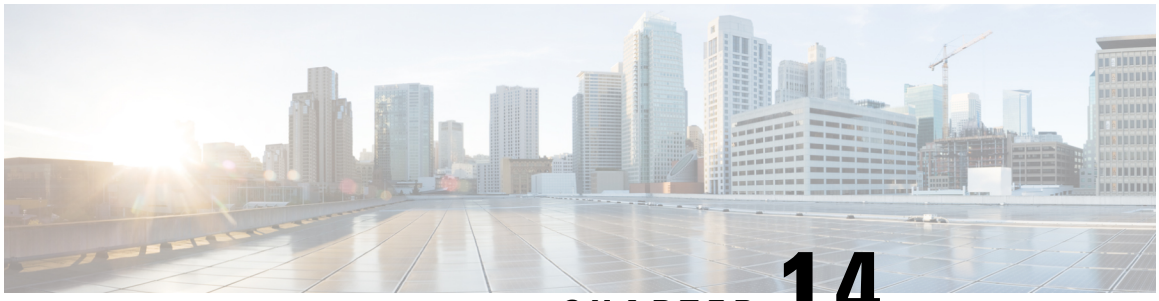
Feature History for MPLS Static Labels

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 16.6.1	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 Cupertino 17.7.1	MPLS Static Labels	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 14

Configuring MPLS Traffic Engineering and Enhancements

Prerequisites for MPLS Traffic Engineering and Enhancements

Ensure that your network supports the following Cisco IOS features before you enable MPLS TE:

- Multiprotocol Label Switching
- IP Cisco Express Forwarding
- Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF)

Restrictions for MPLS Traffic Engineering and Enhancements

- MPLS TE fast reroute is not supported.
- MPLS TE supports only a single IGP process or instance. Multiple IGP processes or instances are not supported and MPLS TE should not be configured in more than one IGP process or instance.
- The MPLS TE feature does not support routing and signaling of LSPs over unnumbered IP address links. Therefore, do not configure the feature over those links.
- When specifying an explicit path, if you specify the *forward* address (the address of the interface that forwards the traffic to the next router) as the next-hop address, the explicit path might not be used. Using the forward address allows that entry to be treated as a loose hop for path calculation. We recommend that you use the *receive* address (the address of the interface that receives traffic from the sending router) as the next-hop address.

In the following example, switch S3 sends traffic to switch S1. The paths marked a,b and x,y between switches S1 and S2 are parallel paths.

```
S1(a)----(b)S2(c)--(d)S3
  (x)----(y)
```

If you configure an explicit path from S3 to S1 using the *forward* addresses (addresses d and b), the tunnel might reroute traffic over the parallel path (x,y) instead of the explicit path. To ensure that the tunnel uses the explicit path, specify the *receive* addresses as part of the **next-address** command, as shown in the following example:

```
ip explicit-path name path1
  next-address (c)
  next-address (a)
```

Information About MPLS Traffic Engineering and Enhancements

The following sections provide information about MPLS TE and enhancements.

Introduction to MPLS Traffic Engineering and Enhancements

MPLS is an integration of Layer 2 and Layer 3 technologies. By making traditional Layer 2 features available to Layer 3, MPLS enables traffic engineering. Thus, you can offer in a one-tier network what can be achieved only by overlaying a Layer 3 network on a Layer 2 network.

Traffic engineering is essential for service provider and ISP backbones. Such backbones must support a high use of transmission capacity, and the networks must be resilient so that they can withstand link or node failures.

MPLS traffic engineering provides an integrated approach to traffic engineering. With MPLS, traffic engineering capabilities are integrated into Layer 3, which optimizes the routing of IP traffic, given the constraints imposed by backbone capacity and topology.

MPLS traffic engineering supports the following functionalities:

- Enhances standard Interior Gateway Protocols (IGPs), such as IS-IS or OSPF, to map packets to the appropriate traffic flows automatically.
- Transports traffic flows across a network using MPLS forwarding.
- Determines the routes for traffic flows across a network. This is based on the resources the traffic flow requires and the resources available in the network.
- Employs Constraint-based routing, in which the path for a traffic flow is the shortest path that meets the resource requirements (constraints) of the traffic flow. In MPLS traffic engineering, the traffic flow has bandwidth requirements, media requirements, a priority that is compared to the priority of other flows, and so forth.
- Recovers from link or node failures by adapting to the new constraints presented by the changed topology.
- Transports packets using MPLS forwarding across a multihop label-switched path (LSP).
- Uses the routing and signaling capability of LSPs across a backbone topology that:
    - Understands the backbone topology and available resources.

- Accounts for link bandwidth and size of traffic flow when determining routes for LSPs across the backbone.
- Has a dynamic adaptation mechanism that enables the backbone to be resilient to failures, even if several primary paths are precalculated offline.
- Includes enhancements to the IGP (IS-IS or OSPF) shortest path first (SPF) calculations to automatically calculate which traffic should be sent over which LSPs.

Benefits of MPLS Traffic Engineering

A WAN connection is an expensive item in an ISP budget. Traffic engineering enables ISPs to route network traffic to offer the best service to their users in terms of throughput and delay. By making the service provider more efficient, traffic engineering reduces the cost of the network.

Currently, some ISPs base their services on an overlay model. In this model, transmission facilities are managed by Layer 2 switching. The routers see only a fully meshed virtual topology, making most destinations appear one hop away. If you use the explicit Layer 2 transit layer, you can precisely control how traffic uses the available bandwidth. However, the overlay model has numerous disadvantages. MPLS traffic engineering achieves the traffic engineering benefits of the overlay model without running a separate network, and without needing a nonscalable, full mesh of router interconnects.

How MPLS Traffic Engineering Works

MPLS TE automatically establishes and maintains LSPs across the backbone by using RSVP. The path that an LSP uses is determined by the LSP resource requirements and network resources, such as bandwidth.

Available resources are flooded by means of extensions to a link state-based IGP.

Traffic engineering tunnels are calculated at the LSP head, based on a fit between the required and available resources (constraint-based routing). The IGP automatically routes the traffic to these LSPs. Typically, a packet traveling across the MPLS TE backbone travels on a single LSP that connects the ingress point to the egress point.

MPLS traffic engineering is built on the following Cisco IOS mechanisms:

- IP tunnel interfaces

  From a Layer 2 standpoint, an MPLS tunnel interface represents the head of an LSP. It is configured with a set of resource requirements, such as bandwidth, media requirements, and priority.

  From a Layer 3 standpoint, an LSP tunnel interface is the headend of a unidirectional virtual link to the tunnel destination.

- MPLS traffic engineering path calculation module

  This calculation module operates at the LSP head. The module determines a path to use for an LSP. The path calculation uses a link-state database containing flooded topology and resource information.

- RSVP with traffic engineering extensions

  RSVP operates at each LSP hop and is used to signal and maintain LSPs based on the calculated path.

- MPLS traffic engineering link management module

This module operates at each LSP hop. It enables link call admission on the RSVP signaling messages, and bookkeeping of topology and resource information to be flooded.

- Link-state IGP (IS-IS or OSPF, each with traffic engineering extensions)

  These IGPs are used to globally flood topology and resource information from the link management module.

- Enhancements to the SPF calculation used by the link-state IGP (IS-IS or OSPF)

  The IGP automatically routes traffic to the appropriate LSP tunnel based on tunnel destination. Static routes can also be used to direct traffic to LSP tunnels.

- Label switching forwarding

  This forwarding mechanism provides routers with a Layer 2-like ability to direct traffic across multiple hops of the LSP established by RSVP signaling.

One approach to engineering a backbone is to define a mesh of tunnels from every ingress device to every egress device. The MPLS TE path calculation and signaling modules determine the path taken by the LSPs for these tunnels, subject to resource availability and the dynamic state of the network. The IGP, operating in an ingress device, determines which traffic should go to which egress device, and steers that traffic into the tunnel from ingress to egress.

A flow from an ingress device to an egress device might be so large that it cannot fit over a single link. And it cannot be carried by a single tunnel. In such a scenario, multiple tunnels between a given ingress and egress can be configured, and the flow load can be shared among them.

Mapping Traffic into Tunnels

This section describes how traffic is mapped to tunnels. It describes how conventional hop-by-hop link-state routing protocols interact with MPLS TE capabilities. This section describes how the shortest path first (SPF) algorithm, sometimes called a Dijkstra algorithm, is enhanced. This enhancement allows a link-state IGP to forward traffic automatically over tunnels that MPLS traffic engineering establishes.

Link-state protocols, such as integrated IS-IS or OSPF, use an SPF algorithm to compute a shortest path tree from the headend node to all the nodes in the network. Routing tables are derived from this shortest path tree. The routing tables contain ordered sets of destination and first-hop information. If a router does normal hop-by-hop routing, the first hop is over a physical interface attached to the router.

New traffic engineering algorithms calculate explicit routes to one or more nodes in the network. The originating router views these explicit routes as logical interfaces. In the context of this document, these explicit routes are represented by LSPs and referred to as traffic engineering tunnels (TE tunnels).

The following sections describe how link-state IGPs can use these shortcuts, and how they can install routes in the routing table that point to these TE tunnels. These tunnels use explicit routes. The path taken by a TE tunnel is controlled by the router that is the headend of the tunnel. In the absence of errors, TE tunnels are guaranteed to not loop, but routers must agree on how to use the TE tunnels. Otherwise, traffic might loop through two or more tunnels.

When specifying an explicit path for an MPLS TE tunnel, you can specify link or node addresses of the next-hop routers in an explicit path. You can also specify a mixture of link and node addresses. There are no restrictions when specifying a mixture of link and node addresses.

Transition of an IS-IS Network to a New Technology

IS-IS, as specified in RFC 1142, includes extensions for MPLS TE. Running MPLS traffic engineering over IS-IS or taking advantage of these other extensions requires transitioning an IS-IS network to this new technology. This section describes these extensions. It discusses two ways in which to migrate an existing IS-IS network from the standard ISO 10589 protocol towards the version of IS-IS specified in RFC 1142. Running MPLS TE over an existing IS-IS network requires a transition to the version of IS-IS specified in RFC 1142. However, running MPLS TE over OSPF does *not* require any similar network transition.

Extensions for the IS-IS Routing Protocol

Extensions for the IS-IS routing protocol serve the following purposes:

- Remove the 6-bit limit on link metrics.
- Allow interarea IP routes.
- Enable IS-IS to carry different kinds of information for traffic engineering. In the future, more extensions might be needed.

To serve these purposes, two new type, length, and value (TLV)objects have been defined:

- TLV 22 describes links (or adjacencies). It serves the same purpose as the IS neighbor option in ISO 10589 (TLV 2).
- TLV 135 describes reachable IP prefixes. It is similar to the IP neighbor options from RFC 1195 (TLVs 128 and 130).

Note For ease of use, these two new TLVs, 22 and 135, are referred to as new-style TLVs and TLVs 2, 128, and 130 are referred to as old-style TLVs.

Both the new TLVs have a fixed length part, followed by optional sub-TLVs. The metric space in these new TLVs has been enhanced from 6 bits to 24 or 32 bits. The sub-TLVs allow you to add new properties to links and prefixes. Traffic engineering is the first technology to use this capability to add new properties to a link.

Solution 1 for Transitioning an IS-IS Network to a New Technology

When you migrate from old-style TLVs towards new-style TLVs, you can advertise the same information twice—once in old-style TLVs and once in new-style TLVs. This ensures that all the devices can understand what is advertised.

There are three disadvantages to using that approach:

- Size of the LSPs: During the transition, the LSPs grow to about twice their original size. This might be a problem in networks where the LSP database (LSPDB) is large. An LSP database might be large because:
  - There are many devices, and therfore, many LSPs.
  - There are many neighbors or IP prefixes per router. A device that advertises lots of information causes the LSPs to be fragmented.

- Unpredictable results: In a large network, this approach can produce unpredictable results. A large network that is in transition pushes the limits with regard to LSP flooding and SPF scaling.
- Ambiguity: If a device encounters different information in the old-style TLVs and the new-style TLVs, it may not be clear what the device should do.
  - You can expect some extra network instability. At this time, you must not test how far you can push an implementation.
  - Traffic engineering extensions might cause LSPs to be reflooded frequently.

Most of these problems can be solved easily by using:

- All the information in the old-style and new-style TLVs in an LSP.
- The adjacency with the lowest link metric if an adjacency is advertised more than once.

The main benefit to advertising the same information twice is that network administrators can use new-style TLVs before all the devices in the network can understand them.

Transition Actions During Solution 1

When transitioning from using IS-IS with old-style TLVs to new-style TLVs, perform the following actions:

- If all the devices run old software, advertise and use only old-style TLVs.
- Upgrade some devices to newer software.
- Configure some devices with new software to advertise both old-style and new-style TLVs. They accept both styles of TLVs. Configure other devices (with old software) to continue advertising and using only old-style TLVs.
- Test traffic engineering in parts of your network.
- If the whole network needs to migrate, upgrade and configure all the remaining devices to advertise and accept both styles of TLVs.
- Configure all the devices to advertise and accept only new-style TLVs.
- Configure metrics larger than 63.

Solution 2 for Transitioning an IS-IS Network to a New Technology

Devices advertise only one style of TLVs at the same time, but can understand both types of TLVs during migration. There are two main benefits to this approach:

- LSPs stay approximately the same size during migration.
- There is no ambiguity when the same information is advertised twice inside one LSP.

This method is useful when you are transitioning the whole network (or a whole area) to use wider metrics (that is, you want a router running IS-IS to generate and accept only new-style TLVs).

The disadvantage is that all devices must understand the new-style TLVs before any router can start advertising new-style TLVs. It does not help the second problem, where network administrators want to use the new-style TLVs for traffic engineering, while some devices are capable of understanding only old-style TLVs.

Transition Actions During Solution 2

If you use the second solution, you can perform the following actions:

- If all the devices run old software, advertise and use only old-style TLVs.
- Upgrade all the devices to newer software.
- Configure all the devices one-by-one to advertise old-style TLVs, but to accept both styles of TLVs.
- Configure all the devices one-by-one to advertise new-style TLVs, but to accept both styles of TLVs.
- Configure all the devices one-by-one to advertise and to accept only new-style TLVs.
- Configure metrics larger than 63.

TLV Configuration Commands

You can use the **metric-style** command to configure the type of TLVs that are accepted by a device. When the device is in IS-IS configuration mode, you can configure the following keywords in the **metric-style** command.

- **metric-style narrow** : Enables the device to generate and accept only old-style TLVs
- **metric-style transition** : Enables the device to generate and accept both old-style and new-style TLVs
- **metric-style wide** : Enables the device to generate and accept only new-style TLVs

You can use either of the following transition schemes when you use the **metric-style** command:

- Narrow to transition to wide.
- Narrow to narrow transition to wide transition to wide.

Implementation in Cisco IOS XE Software

Cisco IOS XE can implement both transition solutions. Network administrators can choose the solution that suits them best. For test networks, solution 1 is best (see Solution 1 for Transitioning an IS-IS Network to a New Technology, on page 205). For a full transition, both solutions can be used. Solution 1 requires fewer steps and less configuration. Solution 2 is for the largest networks, where a risk of doubling the LSP database during transition exists (see Solution 2 for Transitioning an IS-IS Network to a New Technology, on page 206).

How to Configure MPLS Traffic Engineering and Enhancements

The following sections provide information about the steps to configure the MPLS Traffic Engineering and Enhancements feature.

Configuring a Device to Support Tunnels

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** **Example:** `Device(config)# ip cef`	Enables standard Cisco Express Forwarding operation.
Step 4	**mpls traffic-eng tunnels** **Example:** `Device(config)# mpls traffic-eng tunnels`	Enables MPLS traffic engineering tunnels on a device.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits to privileged EXEC mode.

Configuring an Interface to Support RSVP-Based Tunnel Signaling and IGP Flooding

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	**interface** *type slot / subslot / port* [*subinterface-number*] **Example:** `Device(config)# interface Port-channel 114`	Configures an interface type and enters interface configuration mode.
Step 4	**mpls traffic-eng tunnels** **Example:** `Device(config-if)# mpls traffic-eng tunnels`	Enables MPLS traffic engineering tunnels on an interface.
Step 5	**ip rsvp bandwidth** *bandwidth* **Example:** `Device(config-if)# ip rsvp bandwidth 1000`	Enables RSVP on an interface and specifies the amount of bandwidth that will be reserved.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring IS-IS for MPLS Traffic Engineering

Note MPLS traffic engineering supports only a single IGP process or instance. Multiple IGP processes or instances are not supported. MPLS traffic engineering should not be configured in more than one IGP process or instance.

To configure IS-IS for MPLS traffic engineering, perform the following steps.

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	**router isis** **Example:** `Device(config)# router isis`	Enables IS-IS routing and specifies an IS-IS process. The device enters configuration mode.
Step 4	**mpls traffic-eng level** **Example:** `Device(config-router)# mpls traffic-eng level-1`	Turns on MPLS traffic engineering for IS-IS level 1.
Step 5	**mpls traffic-eng level** **Example:** `Device(config-router)# mpls traffic-eng level-2`	Turns on MPLS traffic engineering for IS-IS level 2.
Step 6	**mpls traffic-eng router-id** *type number* **Example:** `Device(config-router)# mpls traffic-eng router-id loopback 0`	Specifies that the traffic engineering router identifier for the node is the IP address associated with interface loopback0.
Step 7	**metric-style wide** **Example:** `Device(config-router)# metric-style wide`	Configures a router to generate and accept only new-style type, length, value objects (TLVs).

Configuring OSPF for MPLS Traffic Engineering

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 ospf** *process-id* **Example:** `Device(config)# router ospf 200`	Configures an OSPF routing process for IP and enters router configuration mode. • The value for the *process-id* argument is an internally used identification parameter for an OSPF routing process. It is locally assigned and can be any positive integer. Assign a unique value for each OSPF routing process.
Step 4	**mpls traffic-eng area** *number* **Example:** `Device(config-router)# mpls traffic-eng area 0`	Turns on MPLS TE for the indicated OSPF area.
Step 5	**mpls traffic-eng router-id loopback0** **Example:** `Device(config-router)# mpls traffic-eng router-id loopback0`	Specifies that the TE router identifier for the node is the IP address associated with interface loopback0.
Step 6	**exit** **Example:** `Device(config-router)# exit`	Exits to global configuration mode.
Step 7	**exit** **Example:** `Device(config)# exit`	Exits to privileged EXEC mode.

Configuring an MPLS Traffic Engineering Tunnel

To configure a preferred explicit path for an MPLS TE tunnel, 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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface tunnel** *number* **Example:** `Device(config)# interface Tunnel0`	Configures an interface type and enters interface configuration mode. • The *number* argument is the number of the tunnel.
Step 4	**ip unnumbered** *type number* **Example:** `Device(config-if)# ip unnumbered loopback0`	Enables IP processing on an interface without assigning an explicit IP address to the interface. • The *type* and *number* arguments name the type and number of another interface on which the router has an assigned IP address. It cannot be another unnumbered interface. • An MPLS traffic engineering tunnel interface should be unnumbered because it represents a unidirectional link.
Step 5	**tunnel destination** *ip-address* **Example:** `Device(config-if)# tunnel destination 192.168.4.4`	Specifies the destination for a tunnel interface. • The *ip-address* argument must be the MPLS traffic engineering router ID of the destination device.
Step 6	**tunnel mode mpls traffic-eng** **Example:** `Device(config-if)# tunnel mode mpls traffic-eng`	Sets the tunnel encapsulation mode to MPLS traffic engineering.
Step 7	**tunnel mpls traffic-eng bandwidth** *bandwidth* **Example:** `Device(config-if)# tunnel mpls traffic-eng bandwidth 250`	Configures the bandwidth for the MPLS traffic engineering tunnel. • The *bandwidth* argument is a number in kilobits per second that is set aside for the MPLS traffic engineering tunnel. Range is from 1 through 4294967295. **Note** If automatic bandwidth is configured for the tunnel, use the **tunnel mpls traffic-eng bandwidth** command to configure the initial tunnel bandwidth.
Step 8	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | **identifier** *path-number*}} [**lockdown**]	Configures the tunnel to use a named IP explicit path or a path dynamically calculated from the traffic engineering topology database.

	Command or Action	Purpose
	Example: `Device(config-if)# tunnel mpls traffic-eng path-option 10 explicit identifier 321`	• The *number* argument is the preference for this path option. When you configure multiple path options, lower numbered options are preferred. Valid values are from 1 through 1000. • The **dynamic** keyword indicates that the path of the LSP is dynamically calculated. • The **explicit** keyword indicates that the path of the LSP is an IP explicit path. • The **name** *path-name* keyword and argument are the path name of the IP explicit path that the tunnel uses with this option. • The **identifier** *path-number* keyword and argument pair names the path number of the IP explicit path that the tunnel uses with this option. The range is from 1 through 65535. • The **lockdown** keyword specifies that the LSP cannot be reoptimized. **Note** A dynamic path is used if an explicit path is currently unavailable.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

Configuring an MPLS Traffic Engineering Tunnel that an IGP Can Use

To configure an MPLS TE tunnel that an IGP can use, perform this procedure.

Procedure

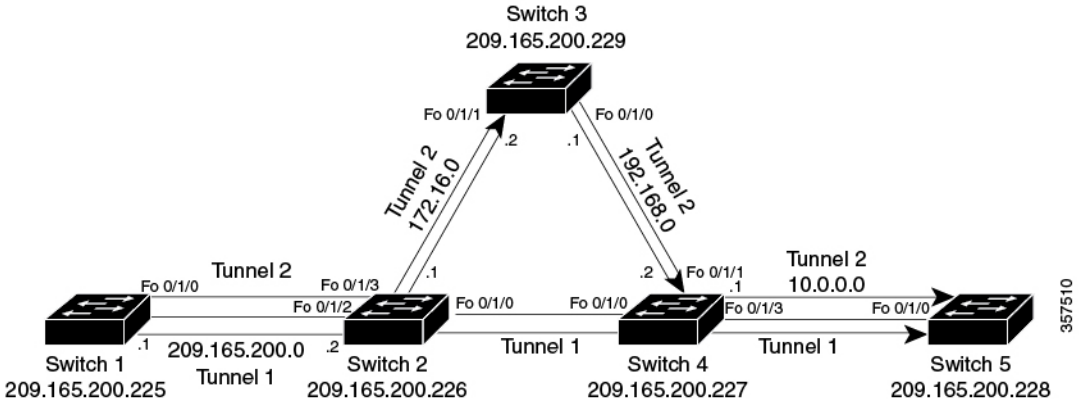
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *number* **Example:** `Device(config)# interface tunnel10`	Configures an interface type and enters interface configuration mode.
Step 4	**ip unnumbered** *type number* **Example:** `Device(config-if)# ip unnumbered loopback 0`	Gives the tunnel interface an IP address. An MPLS traffic engineering tunnel interface should be unnumbered because it represents a unidirectional link.
Step 5	**tunnel destination** *ip-address* **Example:** `Device(config-if)# tunnel destination 10.20.1.1`	Specifies the destination for a tunnel. The *ip-address* keyword is the IP address of the host destination expressed in dotted decimal notation.
Step 6	**tunnel mode mpls traffic-eng** **Example:** `Device(config-if)# tunnel mode mpls traffic-eng`	Sets the tunnel encapsulation mode to MPLS traffic engineering.
Step 7	**tunnel mpls traffic-eng bandwidth** *bandwidth* **Example:** `Device(config-if)# tunnel mpls traffic-eng bandwidth 1000`	Configures the bandwidth for the MPLS traffic engineering tunnel.
Step 8	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name*} | **identifier** *path-number*} [**lockdown**] **Example:** `Device(config-if)# tunnel mpls traffic-eng path-option 1 explicit identifier 1`	Configures the tunnel to use a named IP explicit path or a path dynamically calculated from the traffic engineering topology database. If an explicit path is currently unavailable a dynamic path is used.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

Configuration Examples for MPLS Traffic Engineering and Enhancements

The following figure illustrates a sample MPLS topology. This example specifies point-to-point outgoing interfaces. The subsequent sections contain sample configuration commands that you should enter to implement MPLS traffic engineering and the basic tunnel configuration shown in Figure 3.

Figure 23: Sample MPLS Traffic Engineering Tunnel Configuration



Example: Configuring MPLS Traffic Engineering Using IS-IS

This example lists the commands you should enter to configure MPLS TE with IS-IS routing enabled (see Figure 1).

Note Enter the following commands in every router in the traffic-engineered portion of your network.

Device 1: MPLS Traffic Engineering Configuration

To configure MPLS traffic engineering, enter the following commands:

```
ip cef
mpls traffic-eng tunnels
interface loopback 0
ip address 10.0.0.0 255.255.255.254
ip router isis
interface Fo 1/0/0
ip address 209.165.200.1 255.255.0.0
ip router isis
mpls traffic-eng tunnels
ip rsvp bandwidth 1000
```

Device 1: IS-IS Configuration

To enable IS-IS routing, enter the following commands:

```
router isis
network 47.0000.0011.0011.00
is-type level-1
metric-style wide
mpls traffic-eng router-id loopback0
mpls traffic-eng level-1
```

Example: Configuring MPLS Traffic Engineering Using OSPF

This example lists the commands that you should enter to configure MPLS traffic engineering with OSPF routing enabled (see Figure 1).

Note Enter the following commands in every router in the traffic-engineered portion of your network.

Device 1: MPLS Traffic Engineering Configuration

To configure MPLS traffic engineering, enter the following commands:

```
ip cef
mpls traffic-eng tunnels
interface loopback 0
ip address 209.165.200.225 255.255.255.255
interface Fo 1/0/0
ip address 209.165.200.1 255.255.0.0
mpls traffic-eng tunnels
  ip rsvp bandwidth 1000
```

Device 1: OSPF Configuration

To enable OSPF, enter the following commands:

```
router ospf 0
network 209.165.200.0.0.0.255.255 area 0
mpls traffic-eng router-id Loopback0
mpls traffic-eng area 0
```

Example: Configuring an MPLS Traffic Engineering Tunnel

This example shows you how to configure a dynamic path tunnel and an explicit path in the tunnel. Before you configure MPLS traffic engineering tunnels, enter the appropriate global and interface commands in the specified router (in this case, Router 1).

Device 1: Dynamic Path Tunnel Configuration

To configure a tunnel to use a dynamic path, enter the following commands:

```
interface tunnel1
ip unnumbered loopback 0
tunnel destination 209.165.200.228
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng bandwidth 100
```

```
tunnel mpls traffic-eng priority 1 1
tunnel mpls traffic-eng path-option 1 dynamic
```

Device 1: Dynamic Path Tunnel Verification

To verify that the tunnel is up, enter the following commands:

```
show mpls traffic-eng tunnels
show ip interface tunnel1
```

Device 1: Explicit Path Configuration

To configure an explicit path, enter the following commands:

```
ip explicit-path identifier 1
next-address 209.165.200.1
next-address 172.16.0.1
next-address 192.168.0.1
next-address 10.0.0.1
```

Device 1: Explicit Path Tunnel Configuration

To configure a tunnel to use an explicit path, enter the following commands:

```
interface tunnel2
ip unnumbered loopback 0
tunnel destination 209.165.200.228
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng bandwidth 100
tunnel mpls traffic-eng priority 1 1
tunnel mpls traffic-eng path-option 1 explicit identifier 1
```

Device 1: Explicit Path Tunnel Verification

To verify that the tunnel is up, enter the following commands:

```
show mpls traffic-eng tunnels
show ip interface tunnel2
```

Example: Configuring Enhanced SPF Routing over a Tunnel

This section includes the commands that cause a tunnel to be considered by the IGP's enhanced SPF calculation, that installs routes over the tunnel for appropriate network prefixes.

Device 1: IGP Enhanced SPF Consideration Configuration

To specify that the IGP should use the tunnel (if the tunnel is up) in its enhanced shortest path first (SPF) calculation, enter the following commands:

```
interface tunnel1
tunnel mpls traffic-eng autoroute announce
```

Device 1: Route and Traffic Verification

To verify that the tunnel is up and that the traffic is routed through the tunnel, enter the following commands:

```
#show mpls traffic-eng tunnels tu12001 brief
Signalling Summary:
LSP Tunnels Process: running
Passive LSP Listener: running
RSVP Process: running
Forwarding: enabled
auto-tunnel:
p2p Disabled (0), id-range:62336-64335

Periodic reoptimization: every 3600 seconds, next in 694 seconds
Periodic FRR Promotion: Not Running
Periodic auto-bw collection: every 300 seconds, next in 94 seconds
SR tunnel max label push: 2 primary path labels (2 repair path labels)
TUNNEL NAME DESTINATION UP IF DOWN IF STATE/PROT
tu12001 2.2.2.2 - Po114 up/up
```

Additional References

The following sections provide references related to the MPLS Traffic Engineering and Enhancements feature.

Related Documents

Related Topic	Document Title
IS-IS commands	*Cisco IOS IP Routing Protocols Command Reference*
OSPF command	*Cisco IOS IP Routing Protocols Command Reference*
MPLS TE commands	*Cisco IOS Multiprotocol Label Switching Command Reference*
RSVP commands	*Cisco IOS Quality of Service Solutions Command Reference*

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
1142	*IS-IS*
1195	*Use of OSI IS-IS for Routing in TCP/IP and Dual Environments*
2205	*Resource ReSerVation Protocol (RSVP)*

RFC	Title
2328	*OSPF Version 2*
2370	*The OSPF Opaque LSA Option*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

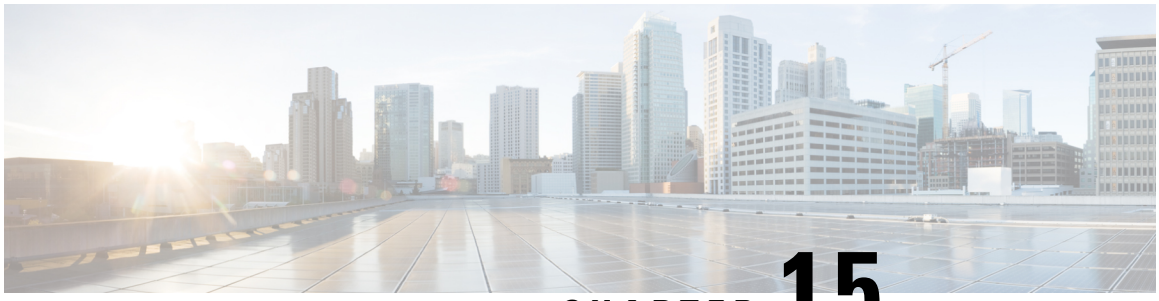
Feature History for MPLS Traffic Engineering and Enhancements

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 Bengaluru 17.6.1	MPLS Traffic Engineering and Enhancements	Multiprotocol Label Switching (MPLS) is an integration of Layer 2 and Layer 3 technologies. By making traditional Layer 2 features available to Layer 3, MPLS enables traffic engineering. Thus, you can offer in a one-tier network what previously could be achieved only by overlaying a Layer 3 network on a Layer 2 network.
Cisco IOS XE Cupertino 17.7.1	MPLS Traffic Engineering and Enhancements	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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/



CHAPTER 15

Configuring Any Transport over MPLS: Tunnel Selection

Restrictions for Any Transport over MPLS: Tunnel Selection

- The **preferred-path** command is available only if the pseudowire encapsulation type is MPLS.
- This feature is enabled when you exit from pseudowire submode.
- The selected path should be a label switched path (LSP) destined to the peer PE router
- The selected tunnel must be an MPLS TE tunnel.
- If you select a tunnel, the tunnel tailend must be on the remote provider edge (PE) router.
- If you specify an IP address, that address must be the IP address of the loopback interface on the remote PE router. The address must have a /32 mask. There must be an LSP destined to that selected address. The LSP need not be a TE tunnel.

Information About Any Transport over MPLS: Tunnel Selection

This feature allows you to specify the path that Any Transport over MPLS (AToM) traffic uses. You can specify either a Multiprotocol Label Switching (MPLS) Traffic Engineering tunnel or a destination IP address and Domain Name System (DNS) name. If the specified path is unreachable, you can specify that the virtual circuits (VCs) should use the default path, which is the path that MPLS Label Distribution Protocol (LDP) used for signaling. This option is enabled by default; you must explicitly disable it.

How to Configure Any Transport over MPLS: Tunnel Selection

The following section provides information about the procedures you can perform to configure Any Transport over MPLS: Tunnel Selection.

Configuring Any Transport over MPLS: Tunnel Selection

You can configure tunnel selection when you set up a pseudowire class. You can enable tunnel selection with the **preferred-path** command. Then you can apply the pseudowire class to an interface that has been configured to transport AToM packets.

To configure Any Transport over MPLS: Tunnel Selection, 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	**psuedowire-class** *name* **Example:** `Device(config)# pseudowire-class ts1`	Establishes a pseudowire class with a name that you specify and enters pseudowire configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-pw)# encapsulation mpls`	Specifies the tunneling encapsulation. For AToM, the encapsulation type is **mpls**.
Step 5	**preferred-path** {**interface-tunnel** *tunnel-number* | **peer** {*ip-address* | *host-name*}} [**disable-fallback**] **Example:** `Device(config-pw)# preferred path peer 10.18.18.18`	Specifies the MPLS traffic engineering tunnel or IP address or host name to be used as the preferred path.
Step 6	**exit** **Example:** `(config-pw)# exit`	Exits pseudowire configuration mode.
Step 7	**interface** *slot/port* **Example:**	Specifies an interface and enters interface configuration mode.

	Command or Action	Purpose
	`Device(config)# interface atm1/1`	
Step 8	**encapsulation** *encapsulation-type* **Example:** `Device(config-if)# encapsulation aal5`	
Step 9	**xconnect** *peer-router-id vcid* **pw-class** *name* **Example:** `Device(config-if)# xconnect 10.0.0.1 123 pw-class ts1`	Binds the attachment circuit to a pseudowire VC.

Configuration Examples for Any Transport over MPLS: Tunnel Selection

The following section provides examples for configuring Any Transport over MPLS: Tunnel Selection.

Example: Configuring Tunnel Selection

The following example sets up two preferred paths for PE1. One preferred path specifies an MPLS traffic engineering tunnel. The other preferred path specifies an IP address of a loopback address on PE2. There is a static route configured on PE1 that uses a TE tunnel to reach the IP address on PE2.

Device PE1

```
mpls label protocol ldp
mpls traffic-eng tunnels
tag-switching tdp router-id Loopback0
pseudowire-class pw1
 encapsulation mpls
 preferred-path interface Tunnel1 disable-fallback
!
interface Loopback0
 ip address 10.2.2.2 255.255.255.255
 no ip directed-broadcast
 no ip mroute-cache
!
interface Tunnel1
 ip unnumbered Loopback0
 no ip directed-broadcast
 tunnel destination 10.16.16.16
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng priority 7 7
 tunnel mpls traffic-eng bandwidth 1500
 tunnel mpls traffic-eng path-option 1 explicit name path-tu1
!
interface gigabitethernet0/0/0
 no ip address
 no ip directed-broadcast
 no negotiation auto
!
```

```
interface gigabitethernet0/0/0.1
 encapsulation dot1Q 222
 no ip directed-broadcast
 xconnect 10.16.16.16 101 pw-class pw1
!
interface gigabitEthernet2/0/1
 ip address 10.0.0.1 255.255.255.0
 no ip directed-broadcast
 tag-switching ip
 mpls traffic-eng tunnels
 ip rsvp bandwidth 15000 15000
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.255 area 0
 network 10.2.2.2 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
!
ip route 10.18.18.18 255.255.255.255 Tunnel2
!
ip explicit-path name path-tu1 enable
 next-address 10.0.0.1
 index 3 next-address 10.0.0.1
```

Device PE2

```
mpls label protocol ldp
mpls traffic-eng tunnels
mpls ldp router-id Loopback0
interface Loopback0
 ip address 10.16.16.16 255.255.255.255
 no ip directed-broadcast
 no ip mroute-cache
!
interface Loopback2
 ip address 10.18.18.18 255.255.255.255
 no ip directed-broadcast
!
interface gigabitEthernet3/1
 ip address 10.0.0.2 255.255.255.0
 no ip directed-broadcast
 mpls traffic-eng tunnels
 mpls ip
 no cdp enable
 ip rsvp bandwidth 15000 15000
!
interface gigabitEthernet3/3
 no ip address
 no ip directed-broadcast
 no cdp enable
!
interface gigabitEthernet3/3.1
 encapsulation dot1Q 222
 no ip directed-broadcast
 no cdp enable
 mpls l2transport route 10.2.2.2 101
!
interface ATM5/0
 no ip address
 no ip directed-broadcast
 no atm enable-ilmi-trap
 no atm ilmi-keepalive
 pvc 0/50 l2transport
```

```
  encapsulation aal5
  xconnect 10.2.2.2 150 encapsulation mpls
!
router ospf 1
 log-adjacency-changes
 network 10.0.0.0 0.0.0.255 area 0
 network 10.16.16.16 0.0.0.0 area 0
 mpls traffic-eng router-id Loopback0
 mpls traffic-eng area 0
```

Example: Verifying the Configuration

In the following example, the **show mpls l2transport vc** command shows the following information (in bold) about the VCs:

- VC 101 has been assigned a preferred path called Tunnel1. The default path is disabled because the preferred path specified that the default path should not be used if the preferred path fails.
- VC 150 has been assigned an IP address of a loopback address on PE2. The default path can be used if the preferred path fails.

```
Device# show mpls l2transport vc detail

Local interface: Gi0/0/0.1 up, Eth VLAN 222 up
    Preferred path: Tunnel1,  active
    Default path: disabled
    Tunnel label: 3, next hop point2point
    Output interface: Tu1, imposed label stack {17 16}
  Create time: 00:27:31, last status change time: 00:27:31
  Signaling protocol: LDP, peer 10.16.16.16:0 up
    MPLS VC labels: local 25, remote 16
    Group ID: local 0, remote 6
    MTU: local 1500, remote 1500
    Remote interface description:
  Sequencing: receive disabled, send disabled
  VC statistics:
    packet totals: receive 10, send 10
    byte totals:   receive 1260, send 1300
    packet drops:  receive 0, send 0
```

Example: Troubleshooting Tunnel Selection

You can use the **debug mpls l2transport vc event** command to troubleshoot tunnel selection. For example, if the tunnel interface that is used for the preferred path is shut down, the default path is enabled. The **debug mpls l2transport vc event** command provides the following output:

```
AToM SMGR [10.2.2.2, 101]: Processing imposition update, vc_handle 62091860, update_action

3, remote_vc_label 16
AToM SMGR [10.2.2.2, 101]: selected route no parent rewrite: tunnel not up
AToM SMGR [10.2.2.2, 101]: Imposition Programmed, Output Interface: Et3/2
```

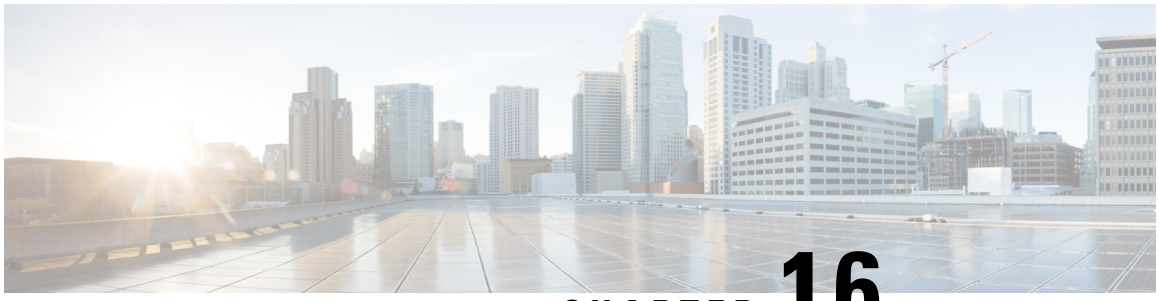
Feature History for Any Transport over MPLS: Tunnel Selection

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 Cupertino 17.7.1	Any Transport over MPLS: Tunnel Selection	The Any Transport over MPLS: Tunnel Selection feature allows you to specify the path that Any Transport over MPLS (AToM) traffic uses. You can specify either a Multiprotocol Label Switching (MPLS) traffic engineering tunnel or a destination IP address and Domain Name System (DNS) name.

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/



CHAPTER 16

Configuring MPLS Traffic Engineering—Bundled Interface Support

Prerequisites for MPLS TE—Bundled Interface Support

- Configure Multiprotocol Label Switching (MPLS) traffic engineering (TE) tunnels.
- Enable Cisco Express Forwarding in global configuration mode.
- Enable Resource Reservation Protocol (RSVP) feature.
- Configure EtherChannel.
- Configure Gigabit EtherChannel.

Restrictions for MPLS TE—Bundled Interface Support

- Traffic engineering over switch virtual interfaces (SVIs) is not supported unless the SVI consists of a bundle of links that represent a single point-to-point interface.
- There must be a valid IP address configuration on the bundled interface and there must not be an IP address configuration on the member links.

Information About MPLS TE—Bundled Interface Support

The MPLS Traffic Engineering—Bundled Interface Support feature enables Multiprotocol Label Switching (MPLS) traffic engineering (TE) tunnels over the bundled interfaces—EtherChannel and Gigabit EtherChannel (GEC).

The Resource Reservation Protocol (RSVP) notifies TE about bandwidth changes that occur when member links are added or deleted, or when links become active or inactive. TE notifies other nodes in the network via Interior Gateway Protocol (IGP) flooding. By default, the bandwidth available to TE Label-Switched Paths (LSPs) is 75 percent of the interface bandwidth. You can change the percentage of the global bandwidth available for TE LSPs by using an RSVP command on the bundled interface. The feature supports bandwidth reservation and preemption.

The following section provides information about Bundled Interface Support for MPLS Traffic Engineering.

Cisco EtherChannel Overview

Cisco EtherChannel technology builds upon standards-based 802.3 full-duplex Fast Ethernet to provide network managers with a reliable, high-speed solution for the campus network backbone. EtherChannel technology provides bandwidth scalability within the campus. It provides up to 800 Mbps, 8 Gbps, or 80 Gbps of aggregate bandwidth for a Fast EtherChannel, Gigabit EtherChannel, or 10 Gigabit EtherChannel connection, respectively. Each of these connection speeds can vary in amounts equal to the speed of the links used (100 Mbps, 1 Gbps, or 10 Gbps). Even in the most bandwidth-demanding situations, EtherChannel technology helps to aggregate traffic, keeps oversubscription to a minimum, and provides effective link-resiliency mechanisms.

Cisco EtherChannel Benefits

Cisco EtherChannel technology allows network managers to provide higher bandwidth among servers, routers, and switches than a single-link Ethernet technology can provide.

Cisco EtherChannel technology provides incremental scalable bandwidth and the following benefits:

- Standards-based—Cisco EtherChannel technology builds upon IEEE 802.3-compliant Ethernet by grouping multiple, full-duplex point-to-point links. EtherChannel technology uses IEEE 802.3 mechanisms for full-duplex autonegotiation and autosensing, when applicable.
- Flexible incremental bandwidth—Cisco EtherChannel technology provides bandwidth aggregation in multiples of 100 Mbps, 1 Gbps, or 10 Gbps. This depends on the speed of the aggregated links. For example, network managers can deploy EtherChannel technology that consists of pairs of full-duplex Fast Ethernet links to provide more than 400 Mbps between the wiring closet and the data center. In the data center, bandwidths of up to 800 Mbps can be provided between servers and the network backbone to provide large amounts of scalable incremental bandwidth.
- Load balancing—Cisco EtherChannel technology comprises several Fast Ethernet links. It is capable of load balancing traffic across those links. Unicast, broadcast, and multicast traffic is evenly distributed across the links, providing improved performance and redundant parallel paths. When a link fails, traffic is redirected to the remaining links within the channel without user intervention and with minimal packet loss.
- Resiliency and fast convergence—When a link fails, Cisco EtherChannel technology provides automatic recovery by redistributing the load across the remaining links. When a link fails, Cisco EtherChannel

technology redirects traffic from the failed link to the remaining links in less than one second. This convergence is transparent to the end user—no host protocol timers expire and no sessions are dropped.

Cisco Gigabit EtherChannel Overview

Cisco Gigabit EtherChannel (GEC) is a high-performance Ethernet technology that provides transmission rates in Gigabit per second (Gbps). A Gigabit EtherChannel bundles individual ethernet links (Gigabit Ethernet and 10 Gigabit Ethernet) into a single logical link. This single link provides the aggregate bandwidth of up to four physical links. All LAN ports in each EtherChannel must be of the same speed and must be configured as either Layer 2 or Layer 3 LAN ports. Broadcast and multicast packets which are inbound on one link in an EtherChannel are blocked from returning on any other link in the EtherChannel.

Load Balancing in EtherChannel

Load balancing affects the actual and practical bandwidth that is in use for TE. Multilink load balancing uses a per-packet load balancing method. The entire bundle interface bandwidth is available. EtherChannel load balancing has various load balancing methods, depending on the traffic pattern and the load balancing configuration. The total bandwidth available for TE may be limited to the bandwidth of a single member link.

How to Configure MPLS TE—Bundled Interface Support

The following section provides information about how to configure Bundled Interface Support for MPLS Traffic Engineering.

Configuring MPLS Traffic Engineering on an EtherChannel Interface

To configure MPLS Traffic Engineering on an etherchannel interface, perform the following 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** *type number* [*name-tag*] **Example:** `Device(config)# interface port-channel 1`	Creates an EtherChannel bundle, assigns a group number to the bundle, and enters interface configuration mode.

	Command or Action	Purpose
Step 4	**ip address** *ip-address mask* [**secondary**] **Example:** `Device(config-if)# ip address 10.0.0.4 255.255.255.0`	Specifies an IP address for the EtherChannel group.
Step 5	**mpls traffic-eng tunnels** **Example:** `Device(config-if)# mpls traffic-eng tunnels`	Enables MPLS TE tunnel signaling on an interface. • Enable MPLS TE tunnel on the device before enabling the signaling.
Step 6	**ip rsvp bandwidth** [*interface-kbps*] [*single-flow-kbps*] **Example:** `Device(config-if)# ip rsvp bandwidth 100`	Enables RSVP for IP on an interface. Specifies a percentage of the total interface bandwidth as available in the RSVP bandwidth pool.
Step 7	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for MPLS Traffic Engineering—Bundled Interface Support

The following section provides configuration examples for MPLS Traffic Engineering—Bundled Interface Support.

Example: Configuring MPLS TE on an EtherChannel Interface

The following example shows how to configure MPLS TE on an EtherChannel interface.

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 1
Device(config-if)# ip address 10.0.0.4 255.255.255.0
Device(config-if)# mpls traffic-eng tunnels
Device(config-if)# ip rsvp bandwidth 100
Device(config-if)# end
```

Example: Configuring MPLS Traffic Engineering—Bundled Interface Support over Gigabit Etherchannel

The following example shows how to enable MPLS Traffic Engineering—Bundled Interface Support over GEC on Cisco devices:

```
Device> enable
Device# configure terminal

! Enable global MPLS TE on routers
Device(config)# router ospf 100
Device(config-router)# network 10.0.0.1 0.0.0.255 area 0
Device(config-router)# mpls traffic-eng area 0
Device(config-router)# mpls traffic-eng router-id Loopback 0
Device(config-router)# exit

! Configure  GEC  interface and enable MPLS TE and RSVP on interface
Device(config)# interface Port-channel 1
Device(config-if)# ip address 10.0.0.1 255.255.255.0
Device(config-if)# mpls traffic-eng tunnels
Device(config-if)# ip rsvp bandwidth
Device(config-if)# exit

! Define explicit path
Device(config)# ip explicit-path name primary enable
Device(cfg-ip-expl-path)# next-address 172.12.1.2
Device(cfg-ip-expl-path)# next-address 172.23.1.2
Device(cfg-ip-expl-path)# next-address 172.34.1.2
Device(cfg-ip-expl-path)# next-address 10.4.4.4
Device(cfg-ip-expl-path)# exit

! Configure primary tunnel on head-end device
Device(config)# interface Tunnel 14
Device(config-if)# ip unnumbered Loopback 0
Device(config-if)# tunnel mode mpls traffic-eng
Device(config-if)# tunnel destination 10.10.10.0
Device(config-if)# tunnel mpls traffic-eng autoroute announce
Device(config-if)# tunnel mpls traffic-eng path-option 10 explicit name primary
Device(config-if)# exit

! Configure GEC interface
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# no ip address
Device(config-if)# channel-group 1 mode active
Device(config-if)# exit

! Configure GEC interface
Device(config)# interface GigabitEthernet 0/0/2
Device(config-if)# no ip address
Device(config-if)# channel-group 1 mode active
Device(config-if)# exit
```

The **show mpls traffic-eng tunnels** command output displays information about a tunnel or one–line information about all tunnels configured on the device:

```
Device# show mpls traffic-eng tunnels tunnel 14

Name: Cat9k_t14                           (Tunnel10) Destination: 10.4.4.4
```

```
  Status:
    Admin: up         Oper: up     Path: valid       Signalling: connected
    path option 1, type explicit toR4overR3R3 (Basis for Setup, path weight 3)

  Config Parameters:
    Bandwidth: 0        kbps (Global)  Priority: 7  7   Affinity: 0x0/0xFFFF
    Metric Type: TE (default)
    AutoRoute: enabled  LockDown: disabled Loadshare: 0 [0] bw-based
    auto-bw: disabled
  Active Path Option Parameters:
    State: explicit path option 1 is active
    BandwidthOverride: disabled  LockDown: disabled  Verbatim: disabled


      InLabel  :  -
  OutLabel : Port-channel1, 1608
  Next Hop : 172.16.1.2
  RSVP Signalling Info:
       Src 10.1.1.1, Dst 10.4.4.4, Tun_Id 14, Tun_Instance 35
    RSVP Path Info:
      My Address: 172.12.1.1
      Explicit Route: 172.12.1.2 172.23.1.1 172.23.1.2 172.34.1.1
                      172.34.1.2 10.4.4.4

  History:
    Tunnel:
      Time since created: 17 hours
      Time since path change: 18 minutes, 22 seconds
      Number of LSP IDs (Tun_Instances) used: 35
    Current LSP: [ID: 35]
      Uptime: 18 minutes, 22 seconds
      Selection: reoptimization
    Prior LSP: [ID: 32]
      ID: path option unknown
      Removal Trigger: signalling shutdown
```

```
Device# show mpls traffic-eng tunnels brief

show mpls traffic-eng tunnels brief
Signalling Summary:
    LSP Tunnels Process:            running
    Passive LSP Listener:           running
    RSVP Process:                   running
    Forwarding:                     enabled
    Periodic reoptimization:        every 3600 seconds, next in 3299 seconds
    Periodic FRR Promotion:         Not Running
    Periodic auto-bw collection:    every 300 seconds, next in 299 seconds

P2P TUNNELS/LSPs:
TUNNEL NAME                      DESTINATION      UP IF     DOWN IF   STATE/PROT^M
Cat9k_t14                      10.4.1.1                      -         Po12      up/up
On Mid Router:
P2P TUNNELS/LSPs:
TUNNEL NAME                      DESTINATION      UP IF     DOWN IF   STATE/PROT
Cat9k_t14                      10.4.1.1                  Po12       Po23       up/up
Cat9k_t23                      10.2.1.1                  Po25        -         up/up
```

Additional References for MPLS Traffic Engineering—Bundled Interface Support

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
MPLS traffic engineering commands	Cisco IOS Multiprotocol Label Switching Command Reference
IPv6 commands	IPv6 Command Reference

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

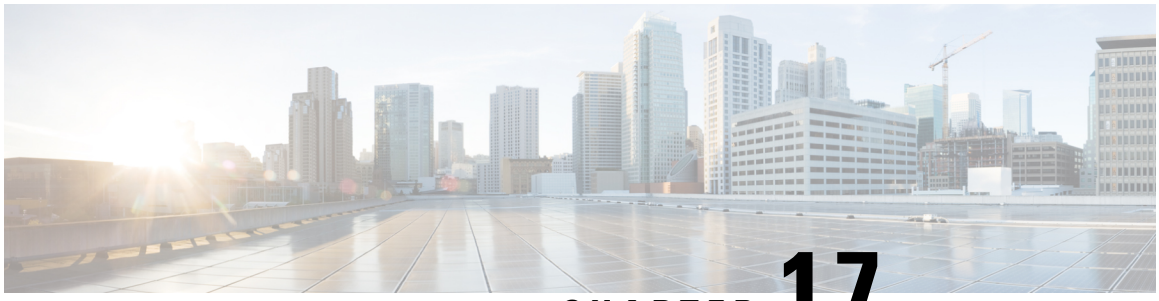
Feature History for MPLS Traffic Engineering—Bundled Interface Support

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 Bengaluru 17.6.1	MPLS Traffic Engineering—Bundled Interface Support	The MPLS Traffic Engineering—Bundled Interface Support feature enables MPLS Traffic Engineering tunnels over the bundled interfaces EtherChannel and Gigabit EtherChannel (GEC).

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/



CHAPTER 17

Configuring MPLS Traffic Engineering Forwarding Adjacency

Prerequisites for MPLS Traffic Engineering Forwarding Adjacency

Your network must support the following Cisco IOS XE features:

- Multiprotocol Label Switching (MPLS)
- IP Cisco Express Forwarding
- IS-IS

Restrictions for MPLS Traffic Engineering Forwarding Adjacency

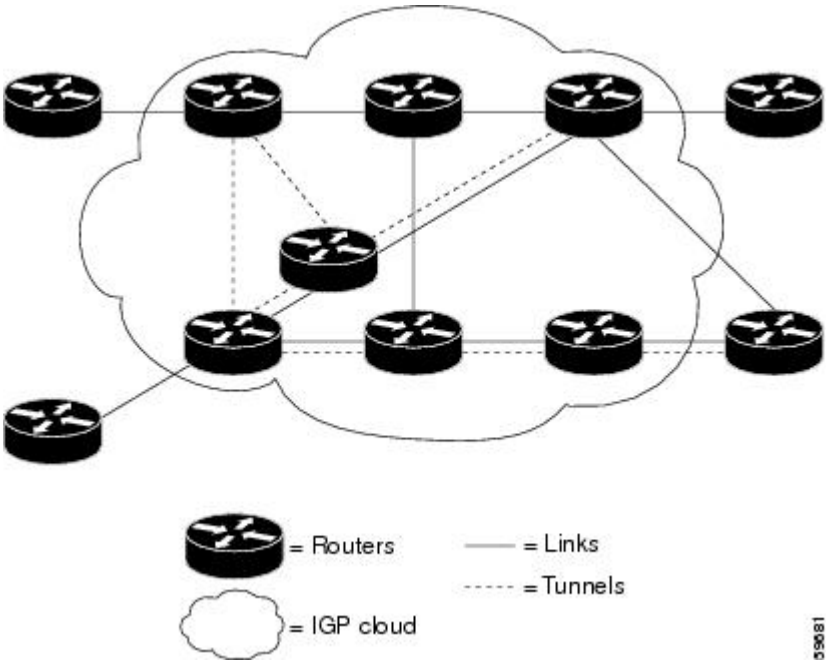
- Using the MPLS Traffic Engineering Forwarding Adjacency feature increases the size of the IGP database by advertising a TE tunnel as a link.
- When the MPLS Traffic Engineering Forwarding Adjacency feature is enabled on a TE tunnel, the link is advertised in the IGP network as a type, length, value (TLV) 22 object without any TE sub-TLV.
- You must configure MPLS TE forwarding adjacency tunnels bidirectionally.

Information About MPLS Traffic Engineering Forwarding Adjacency

The following topics provide information about MPLS Traffic Engineering Forwarding Adjacency.

MPLS Traffic Engineering Forwarding Adjacency Functionality

The MPLS Traffic Engineering Forwarding Adjacency feature allows you to handle a TE LSP tunnel as a link in an IGP network based on the SPF algorithm. A forwarding adjacency can be created between devices regardless of their location in the network. The devices can be located multiple hops from each other, as shown in the figure below.



As a result, a TE tunnel is advertised as a link in an IGP network with the link's cost associated with it.

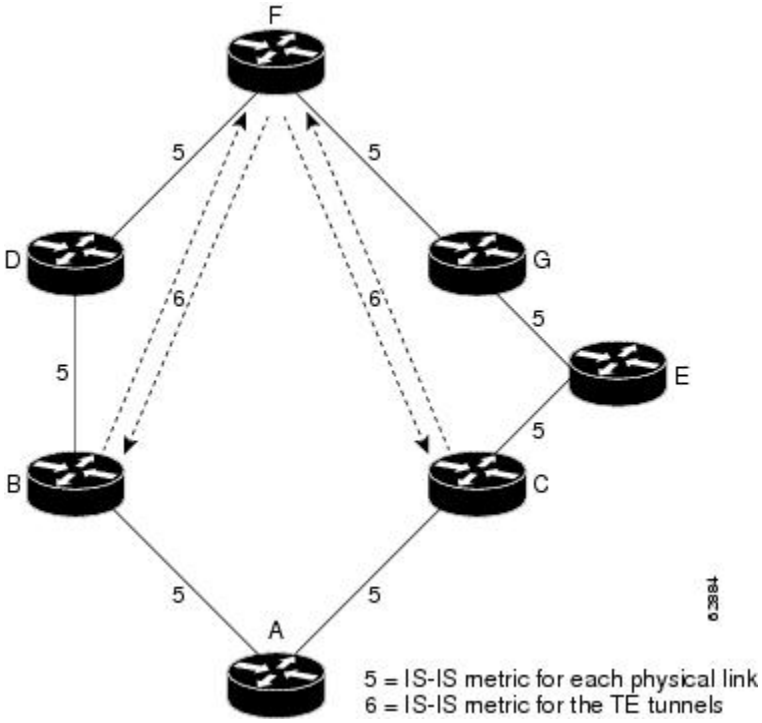
Devices outside of the TE domain see the TE tunnel and use it to compute the shortest path for routing traffic throughout the network.

MPLS Traffic Engineering Forwarding Adjacency Benefits

TE tunnel interfaces advertised for SPF--TE tunnel interfaces are advertised in the IGP network just like any other links. Devices can then use these advertisements in their IGPs to compute the SPF even if they are not the headend of any TE tunnels.

Usage Tips

In the figure below, if you have no forwarding adjacencies configured for the TE tunnels between B and F and C and F, all the traffic that A must forward to F goes through B because B is the shortest path from A to F. (The cost from A to F is 15 through B and 20 through C.)



If you have forwarding adjacencies configured on the TE tunnels between B and F and C and F and also on the TE tunnels between F and B and F and C, then when A computes the SPF algorithm, A sees two equal cost paths of 11 to F. As a result, traffic across the A-B and A-C links is shared.

How to Configure MPLS Traffic Engineering Forwarding Adjacency

The following section provides information about the configuration steps for configuring MPLS Traffic Engineering Forwarding Adjacency.

Configuring a Tunnel Interface for MPLS TE Forwarding Adjacency

To configure a tunnel interface for MPLS TE Forwarding Adjacency, 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:** `Devcie# configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel** *number* **Example:** `Device(config)# interface tunnel 0`	Designates a tunnel interface for the forwarding adjacency, and enters interface configuration mode.
Step 4	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring MPLS TE Forwarding Adjacency on Tunnels with ISIS

To configure MPLS TE Forwarding Adjacency on tunnels with ISIS, perform this procedure.

Note You must configure a forwarding adjacency on two LSP tunnels bidirectionally, from A to B and B to A. Otherwise, the forwarding adjacency is advertised, but not used in the IGP network.

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	**interface tunnel** *number* **Example:** `Device(config)# interface tunnel 0`	Designates a tunnel interface for the forwarding adjacency, and enters interface configuration mode.
Step 4	**ip router isis** *area-tag* **Example:** `Device(config-if)# ip router isis 1`	Configures an IS-IS routing process for IP on an interface and to attaches an area designator to the routing process.
Step 5	**tunnel mpls traffic-eng forwarding-adjacency** [**holdtime** *value*] **Example:** `Device(config-if)# tunnel mpls traffic-eng forwarding-adjacency`	Advertises a TE tunnel as a link in an IGP network.
Step 6	**isis metric** {*metric-value*| **maximum**} {**level-1**| **level-2**} **Example:** `Device(config-if)# isis metric 2 level-1`	Configures the IS-IS metric for a tunnel interface to be used as a forwarding adjacency. • You should specify the **isis metric** command with level-1 or level-2 to be consistent with the IGP level at which you are performing traffic engineering. Otherwise, the metric has the default value of 10.

Configuring MPLS TE Forwarding Adjacency on Tunnels with OSPF

To configure MPLS TE Forwarding Adjacency on tunnels with OSPF, perform this procedure.

Note You must configure a forwarding adjacency on two LSP tunnels bidirectionally, from A to B and B to A. Otherwise, the forwarding adjacency is advertised, but not used in the IGP network.

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 tunnel** *number* **Example:** `Device(config)# interface tunnel 0`	Designates a tunnel interface for the forwarding adjacency, and enters interface configuration mode.
Step 4	**ip ospf** *process-id* **area** *area-id* **Example:** `Device(config-if)# ip router ospf 1 area 0`	Configures an OSPF routing process for IP on an interface and to attaches an area designator to the routing process.
Step 5	**tunnel mpls traffic-eng forwarding-adjacency** [**holdtime** *value*] **Example:** `Device(config-if)# tunnel mpls traffic-eng forwarding-adjacency`	Advertises a TE tunnel as a link in an IGP network.
Step 6	**ip ospf cost** *cost* **Example:** `Device(config-if)# ip ospf cost 4`	Configures the OSPF metric for a tunnel interface to be used as a forwarding adjacency.

Verifying MPLS TE Forwarding Adjacency

To verify MPLS TE Forwarding Adjacency, perform this procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Use this command to enter the privileged EXEC mode. Enter your password, if prompted.
Step 2	**show mpls traffic-eng forwarding-adjacency** [*ip-address*] **Example:** `Device# show mpls traffic-eng forwarding-adjacency`	Use this command to display the current tunnels.

	Command or Action	Purpose
Step 3	**show isis** [*process-tag*] **database** [**level-1**] [**level-2**] [**l1**] [**l2**] [**detail**] [**lspid**] **Example:** `Device# show isis database` **Example:** Use this command to display information about the IS-IS link-state database. For example:	
Step 4	**exit** **Example:** `Device# exit`	Use this command to exit to user EXEC mode.

Configuration Examples for MPLS Traffic Engineering Forwarding Adjacency

This section provides a configuration example for the MPLS Traffic Engineering Forwarding Adjacency feature using an IS-IS metric.

Example MPLS TE Forwarding Adjacency

The following output shows the configuration of a tunnel interface, a forwarding adjacency, and an IS-IS metric:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# interface tunnel 7
Device(config-if)# ip router isis 1
Device(config-if)# tunnel mpls traffic-eng forwarding-adjacency
Device(config-if)# isis metric 2 level-1
```

Following is sample command output when a forwarding adjacency has been configured:

```
Device# show running-config
Building configuration...
Current configuration :364 bytes
!
interface Tunnel7
ip router isis 1
ip unnumbered Loopback0
no ip directed-broadcast
tunnel destination 192.168.1.7
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng forwarding-adjacency
tunnel mpls traffic-eng priority 7 7
tunnel mpls traffic-eng path-option 10 explicit name short
isis metric 2 level 1
```

Note Do not specify the **tunnel mpls traffic-eng autoroute announce** command in your configuration when you are using forwarding adjacency.

Following is an example where forwarding adjacency is configured with OSPF:

```
Device# configure terminal

Device# show running-config

Building configuration...
Current configuration : 310 bytes
interface Tunnel1
ip router ospf 1 area 0
ip unnumbered Loopback0
ip ospf cost 6
tunnel destination 172.16.255.5
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng forwarding-adjacency  tunnel mpls
traffic-eng priority 7 7
tunnel mpls traffic-eng bandwidth 1000
tunnel mpls traffic-eng path-option 10 dynamic
end
Device# show mpls traffic-eng forwarding-adjacency

 destination 172.16.255.5, area ospf 172  area 0, has 1 tunnels
  Tunnel1      (load balancing metric 2000000, nexthop 172.16.255.5)
               (flags:  Forward-Adjacency, holdtime 0)
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
MPLS traffic engineering commands	*Cisco IOS Multiprotocol Label Switching Command Reference*
IP switching commands	*Cisco IOS IP Switching Command Reference*
IS-IS TLVs	Intermediate System-to-Intermediate System (IS-IS) TLVs (white paper)

Standards

Standard	Title
No new or modified standards are supported by this feature, and 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 standards 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

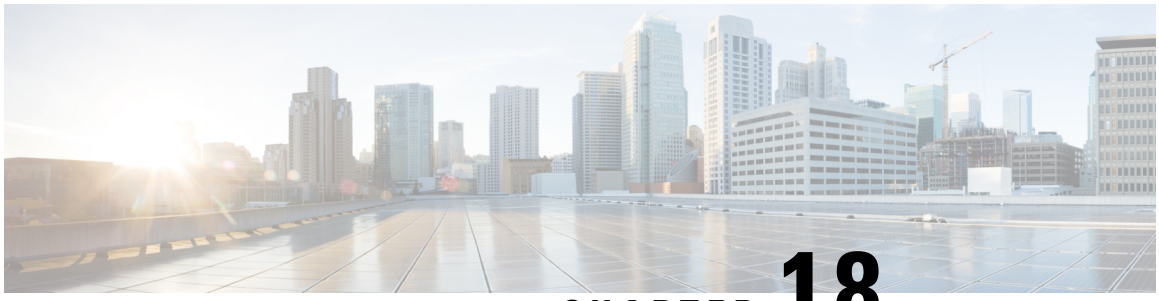
Feature History for MPLS Traffic Engineering Forwarding Adjacency

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 Cupertino 17.7.1	MPLS Traffic Engineering Forwarding Adjacency	The MPLS Traffic Engineering Forwarding Adjacency feature allows a network administrator to handle a traffic engineering (TE) label switched path (LSP) tunnel as a link in an Interior Gateway Protocol (IGP) network based on the Shortest Path First (SPF) algorithm.

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/



CHAPTER 18

Configuring MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion

Prerequisites for MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion

Your network must support the following Cisco IOS features in order to support IP explicit address exclusion:

- MPLS
- IP Cisco Express Forwarding
- Intermediate System-to-Intermediate System (IS-IS) or Open Shortest Path First (OSPF)

Restrictions for MPLS Traffic Engineering (TE)--IP Explicit Address Exclusion

MPLS TE will accept an IP explicit path comprised of either all excluded addresses configured by the **exclude-address** command or all included addresses configured by the **next-address** command. It will not accept a combination of both.

Information About MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion

The MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion feature provides a means to exclude a link or node from the path for a Multiprotocol Label Switching (MPLS) TE label switched path (LSP).

The feature is enabled through the **ip explicit-path** command that allows you to create an IP explicit path and enter a configuration submode for specifying the path. The feature adds to the submode commands the **exclude-address** command for specifying addresses to exclude from the path.

If the excluded address for an MPLS TE LSP identifies a flooded link, the constraint-based shortest path first (CSPF) routing algorithm does not consider that link when computing paths for the LSP. If the excluded address specifies a flooded MPLS TE router ID, the CSPF routing algorithm does not allow paths for the LSP to traverse the node identified by the device ID.

MPLS Traffic Engineering

MPLS is an Internet Engineering Task Force (IETF)-specified framework that provides for the efficient designation, routing, forwarding, and switching of traffic flows through the network.

Traffic engineering (TE) is the process of adjusting bandwidth allocations to ensure that enough is left for high-priority traffic.

In MPLS TE, the upstream device creates a network tunnel for a particular traffic stream, then fixes the bandwidth available for that tunnel.

Cisco Express Forwarding

Cisco Express Forwarding is an advanced, Layer 3 switching technology inside a device. It defines the fastest method by which a Cisco device forwards packets from ingress to egress interfaces. The **ip cef** command enables Cisco Express Forwarding globally, and the **ip route-cache cef** command enables Cisco Express Forwarding on an interface.

How to Configure MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion

The following section provides information about the various configuration steps for MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion.

Configuring IP Explicit Address Exclusion

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 explicit-path** {**name** *path-name* | **identifier** *number*} [**enable** | **disable**] **Example:** `Device(config)# ip explicit-path name OmitR12`	Specifies the name or number of the explicit path, and enables the path, and enters explicit-path configuration mode.
Step 4	**exclude-address** *ip-address* **Example:** `Device(cfg-ip-expl-path)# exclude-address 10.12.12.12`	Excludes the specified link or node from consideration by the constraint-based SPF. • The *ip-address* is a link address or the router ID for a node.
Step 5	**exit** **Example:** `Device(cfg-ip-expl-path)# exit`	Exits from explicit-path configuration mode, and returns to global configuration mode.
Step 6	**exit** **Example:** `Device(config)# exit`	Exits from global configuration mode, and returns to privileged EXEC mode.
Step 7	**show ip explicit-path** **Example:** `Device# show ip explicit-path`	Displays information about configured IP explicit paths.

Configuring an MPLS Traffic Engineering 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** *number* **Example:** `Device(config)# interface tunnel11`	Configures an interface type and enters interface configuration mode.
Step 4	**ip unnumbered loopback0** **Example:** `Device(config-if)# ip unnumbered loopback0`	Assigns the tunnel interface an IP address. • An MPLS traffic engineering tunnel interface should be unnumbered because it represents a unidirectional link.
Step 5	**tunnel destination** *ip-address* **Example:** `Device(config-if)# tunnel destination 10.11.11.11`	Specifies the destination for a tunnel. • The destination of the tunnel must be the MPLS traffic engineering router ID of the destination device.
Step 6	**tunnel mode mpls traffic-eng** **Example:** `Device(config-if)# tunnel mode mpls traffic-eng`	Sets the tunnel encapsulation mode to MPLS traffic engineering.
Step 7	**tunnel mpls traffic-eng bandwidth** *bandwidth* **Example:** `Device(config-if)# tunnel mpls traffic-eng bandwidth 100`	Configures the bandwidth for the MPLS traffic engineering tunnel.
Step 8	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | **ID** *path-number*}} [*lockdown*]	Configures the tunnel to use a named IP explicit path or a path dynamically calculated from the traffic engineering topology database.

	Command or Action	Purpose
	Example: `Device(config-if)# tunnel mpls traffic-eng path-option 2 dynamic`	• If an explicit path is unavailable a dynamic path is used. **Note** To configure a path option that specifies an exclude address, specify the **explicit** keyword (not the **dynamic** keyword) and specify an IP explicit path.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits from interface configuration mode.
Step 10	**exit** **Example:** `Device(config)# exit`	Exits to privileged EXEC mode.
Step 11	**show mpls traffic eng tunnels** **Example:** `Device# show mpls traffic eng tunnels`	Shows information about tunnels, including the current tunnel path when a tunnel is operational. • By viewing the command output, you can determine the path that was used to build a tunnel. If you entered the **exclude-address** command, the specified link or node should not be listed.

Configuration Examples for MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion

The following section provides configuration examples for MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion.

Example: Configuring IP Explicit Address Exclusion

The following example shows how to configure an MPLS TE tunnel with two path options: a preferred explicit path with an excluded address and a backup dynamic path.

Configure the IP explicit path named OmitR12, which excludes the router with router ID 10.12.12.12:

```
ip explicit-path name OmitR12
exclude-address 10.12.12.12
Explicit Path name OmitR12:
```

```
1: exclude-address 10.12.12.12
exit
```

To verify the configuration of the explicit path, use the **show ip explicit-path** command.

```
show ip explicit-paths name OmitR12
PATH OmitR12 (loose source route, path complete, generation 3)
 1: exclude-address 10.12.12.12
```

Note You must know the router IDs for LSRs (nodes) in the network; in this example, that 10.12.12.12 is a router ID. Otherwise, it will not be apparent whether the specified address is the IP address of a link or a router ID.

Example: Configuring an MPLS Traffic Engineering Tunnel

The following example configures Tunnel11 with its two options, where the preferred path option is the IP explicit path OmitR2:

```
interface tunne l11
ip unnumbered loopback0
tunnel destination 10.11.11.11
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng path-option 1 explicit name OmitR12
tunnel mpls traffic-eng path-option 2 dynamic
```

Note There are additional commands for configuring properties for TE tunnels such as bandwidth and priority. For descriptions of those commands, refer to the *Cisco IOS IP Switching Services Configuration Guide* .

Additional References

The following sections provide references related to the MPLS Traffic Engineering (TE) – IP Explicit Address Exclusion feature.

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
MPLS commands	*Cisco IOS Multiprotocol Label Switching Command Reference*
MPLS configuration information	*Cisco IOS XE Multiprotocol Label Switching Configuration Guide*

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 IOS 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 website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

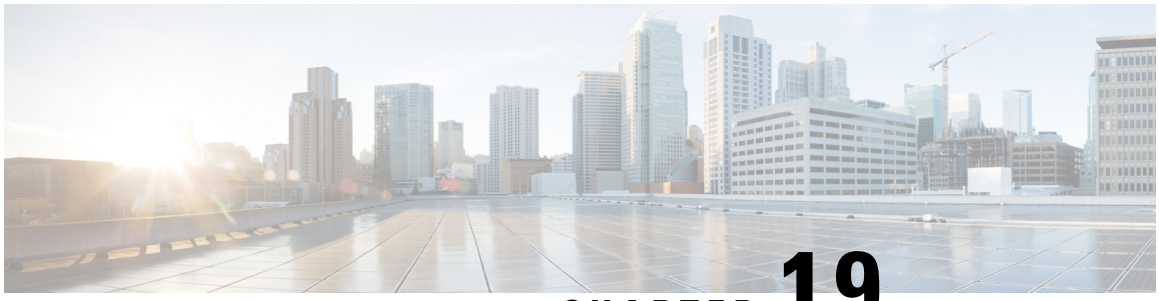
Feature History for MPLS Traffic Engineering (TE)IP—Explicit Address Exclusion

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 Bengaluru 17.6.1	MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion	The MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion feature provides a means to exclude a link or node from the path for Multiprotocol Label Switching (MPLS) TE label switched path (LSP).
Cisco IOS XE Cupertino 17.7.1	MPLS Traffic Engineering (TE)—IP Explicit Address Exclusion	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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/



CHAPTER 19

Configuring MPLS Traffic Engineering—LSP Attributes

Prerequisites for MPLS Traffic Engineering—LSP Attributes

The MPLS Traffic Engineering—LSP Attributes feature requires that you configure an MPLS TE tunnel before you configure either an LSP Attribute List or a Path Option for Bandwidth Override feature.

Restrictions for MPLS Traffic Engineering—LSP Attributes

- Reoptimization between path options with different priorities is not supported.
- With the LSP Attribute List feature, you need to configure priority for path options that is consistent with the priority configured on the tunnel or in other path options used by the tunnel.

Information About MPLS Traffic Engineering—LSP Attributes

The following section provides information about MPLS Traffic Engineering—LSP Attributes.

MPLS Traffic Engineering—LSP Attributes

This document describes how to configure label switched path (LSP) attributes for path options associated with Multiprotocol Label Switching (MPLS) traffic engineering (TE) tunnels.

The MPLS Traffic Engineering—LSP Attributes feature is an extension to MPLS TE. It provides an LSP Attribute list feature and a Path Option for Bandwidth Override feature. These features provide flexibility in the configuration of LSP attributes for MPLS TE tunnel path options. Several LSP attributes can be applied to path options for TE tunnels using an LSP attribute list. If bandwidth is the only LSP attribute you require, then you can configure a Path Option for Bandwidth Override.

MPLS Traffic Engineering—LSP Attributes Benefits

The MPLS Traffic Engineering—LSP Attributes feature provides an LSP Attribute List feature and a Path Option for Bandwidth Override feature. These features have the following benefits:

- The LSP Attributes List feature enables you to configure values for several LSP-specific path options for TE tunnels.
- One or more TE tunnels can specify specific path options by referencing an LSP Attribute List.
- LSP attribute lists make the MPLS TE user interface more flexible, easier to use, and easier to extend and maintain.
- The Path Option for Bandwidth Override feature provides a single command that allows a TE tunnel to fall back temporarily to path options that can reduce bandwidth constraints.

Traffic Engineering Bandwidth

MPLS traffic engineering allows constraint-based routing (CBR) of IP traffic. One of the constraints satisfied by CBR is the availability of required bandwidth over a selected path. Regular TE tunnel bandwidth is called the global pool. Subpool bandwidth is a portion of the global pool. If the subpool bandwidth is not in use it is not reserved from the global pool. Therefore, subpool tunnels require a higher priority than other tunnels.

You can configure the LSP Attribute bandwidth path option to use either global pool (default) or subpool bandwidth. The bandwidth value for the path option may be any valid value. The pool does not have to be the same as that configured on the tunnel.

Note When you configure bandwidth for path options with the **bandwidth** [**global**] *kbps* command, use either all subpool bandwidths or all global-pool bandwidths.

You can configure bandwidth on both dynamic and explicit path options using either the LSP Attribute List feature or the Path Option for Bandwidth Override feature. The commands that enable these features are exclusive of each other. If bandwidth is the only LSP attribute that you need to set on the path option, then use the command to enable the feature. This is the simplest way to configure multiple path options with decreasing bandwidth constraints. Once the **bandwidth** keyword is entered on the **tunnel mpls traffic-eng path-option** command in interface configuration mode, you cannot configure an LSP Attribute List for that path option.

Tunnel Attributes and LSP Attributes

Cisco IOS XE tunneling interfaces have many parameters associated with MPLS TE. Typically, you configure these parameters with **tunnel mpls traffic-eng** commands in interface configuration mode. Many of these commands determine tunnel-specific properties, such as the load-sharing factor for the tunnel. These commands

configure parameters that are unrelated to the particular LSP in use by the tunnel. However, some of the tunneling parameters apply to the LSP that the tunnel uses. You can configure the LSP-specific properties using an LSP Attribute list.

LSP Attributes and the LSP Attribute List

An LSP Attribute list can contain values for each LSP-specific parameter that is configurable for a TE tunnel. You configure an LSP attribute list with the **mpls traffic-eng lsp attributes** *string* command, where *string* identifies the attribute list. The LSP attributes that you can specify include the following:

- Attribute flags for links that make up the LSP (**affinity** command)
- LSP bandwidth--global pool or subpool (**bandwidth** command)
- Disable reoptimization of the LSP (**lockdown** command)
- LSP priority (**priority** command)
- Record the route used by the LSP (**record-route** command)

LSP Attribute Lists Management

The MPLS Traffic Engineering—LSP Attributes feature also provides commands that help you manage LSP Attribute lists. You can do the following:

- Relist all attribute list entries (**list** command)
- Remove a specific attribute from the list (**no***attribute* command)

The **exit** command exits from the LSP attributes configuration submode and returns you to global configuration mode.

Based on your requirements, you can configure LSP attributes lists with different sets of attributes for different path options. LSP attribute lists also provide an easy way to configure multiple TE tunnels to use the same LSP attributes. That is, you can reference the same LSP attribute list to configure LSP-specific parameters for one or more TE tunnels.

Constraint-Based Routing and Path Option Selection

MPLS traffic engineering automatically establishes and maintains LSPs across the backbone by using the Resource Reservation Protocol (RSVP). The path that an LSP uses is determined by the LSP resource requirements and network resources, such as bandwidth. Traffic engineering tunnels are calculated at the LSP head based on a fit between required and available resources (constraint-based routing).

Without the Path Option for Bandwidth Override feature, a TE tunnel establishes an LSP based on dynamic or explicit path options in order of preference. However, the bandwidth and other attributes configured on the TE tunnel allow the setup of an LSP only if LSP path options satisfy the constraints. If a path cannot be found that satisfies the configured path options, then the tunnel is not set up.

The Path Option for Bandwidth Override feature provides a fallback path option that allows overriding the bandwidth configured on the TE tunnel interface. For example, you can configure a path option that sets the bandwidth to zero (0) effectively removing the bandwidth constraint imposed by the constraint-based routing calculation.

Tunnel Reoptimization and Path Option Selection

Reoptimization occurs when a device with traffic engineering tunnels periodically examines tunnels with established LSPs to learn if better LSPs are available. If a better LSP seems to be available, the device attempts to signal the better LSP. If the signaling is successful, the device replaces the older LSP with the new, better LSP.

Reoptimization can be triggered by a timer, the issuance of an **mpls traffic-eng reoptimize** command, or a configuration change that requires the signalling of a tunnel. The MPLS AutoBandwidth feature, for example, uses a timer to set the frequency of reoptimization based on the bandwidth path option attribute. The Path Option for Bandwidth Override feature allows for the switching between bandwidth configured on the TE tunnel interface and bandwidth configured on a specific path option. This increases the success of signaling an LSP for the TE tunnel.

With bandwidth override configured on a path option, the traffic engineering software attempts to reoptimize the bandwidth every 30 seconds to reestablish the bandwidth configured on the tunnel (see the Configuring a Path Option for Bandwidth Override section).

You can disable reoptimization of an LSP with the **lockdown** command in an LSP Attribute list. You can apply the LSP Attribute list containing the **lockdown** command to a path option with the **tunnel mpls traffic-eng path-option** command.

Note When you configure bandwidth for path options with the **bandwidth** [**global**] *kpbs* command, use either all subpool bandwidths or all global-pool bandwidths. Do not mix subpool and other bandwidths, otherwise the path option does not reoptimize later.

Path Option Selection with Bandwidth Override

The Path Option for Bandwidth Override feature allows you to configure bandwidth parameters on a specific path option. The **tunnel mpls traffic-eng path-option** command's **bandwidth** keyword can be used for this purpose. When an LSP is signaled using a path option with a configured bandwidth, the bandwidth associated with the path option is signaled instead of the tunnel's configured bandwidth.

This feature also provides the ability to configure multiple path options that reduce the bandwidth constraint each time the headend of a tunnel fails to establish an LSP.

The following configuration uses the **tunnel mpls traffic-eng bandwidth** command to configure the bandwidth of the tunnel and three **tunnel mpls traffic-eng path-option** commands that define the signalling path options for the LSP:

```
tunnel mpls traffic-eng bandwidth 1000
tunnel mpls traffic-eng path-option 1 explicit name path1
tunnel mpls traffic-eng path-option 2 explicit name path2 bandwidth 500
tunnel mpls traffic-eng path-option 3 dynamic bandwidth 0
```

The device selects a path option for an LSP in order of preference, as follows:

- The device attempts to signal an LSP using path options starting with path option 1.

The device attempts to signal an LSP with the 1000 kbps bandwidth configured on the tunnel interface because path-option 1 has no bandwidth configured.

- If 1000 kbps bandwidth is not available over the network, the device attempts to establish an LSP using path-option 2.

Path option 2 has a bandwidth of 500 kbps configured. This reduces the bandwidth constraint from the original 1000 kbps configured on the tunnel interface.

- If 500 kbps is not available, the device attempts to establish an LSP using path-option 3.

Path-option 3 is configured as dynamic and has bandwidth 0. The device establishes the LSP if an IP path exists to the destination and all other tunnel constraints are met.

Default Path Option Attributes for TE Tunnels Using LSP Attribute Lists

Values for path option attributes for a TE tunnel are determined in this manner:

- LSP attribute list values referenced by the path option take precedence over the values configured on the tunnel interface.
- If you do not specify an attribute in the LSP attribute list, the device uses the attribute in the tunnel configuration. LSP attribute lists do not have defaults.
- If you do not configure the attribute on the tunnel, then the device uses the tunnel default value, as follows:

{Affinity= affinity 0 mask 0,

Bandwidth= bandwidth 0,

Lockdown= no lockdown,

Priority= priority 7 7,

Record-route= no record-route

.

.

.

}

How to Configure MPLS Traffic Engineering—LSP Attributes

The following section provides information on configuring MPLS Traffic Engineering—LSP Attributes.

Configuring an LSP Attribute List

Perform this task to configure a label switched path (LSP) attribute list with the desired attributes to be applied on a path option. Based on your requirements, you can configure LSP attributes lists with different sets of attributes for different path options. The LSP attribute list provides a user interface that is flexible, easy to use, and easy to extend and maintain for the configuration of MPLS TE tunnel path options.

LSP attribute lists also provide an easy way to configure multiple TE tunnels to use the same LSP attributes. That is, you can reference the same LSP attribute list to configure LSP-specific parameters for one or more TE tunnels.

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 traffic-eng lsp attributes** *string* **Example:** `Device(config)# mpls traffic-eng lsp attributes 1`	Configures an LSP attribute list and enters LSP Attributes configuration mode. • The *string* argument identifies a specific LSP attribute list.

Step 4	**affinity** *value* [**mask** *value*]
	Example:
	`Device(config-lsp-attr)# affinity 0 mask 0`
		(Optional) Specifies attribute flags for links comprising an LSP.
		• The *value* argument is a value required for links that make up an LSP. Values of the bits are either 0 or 1.
		• The **mask** *value* keyword argument combination indicates which attribute values should be checked.
		  • If a bit in the mask is 0, an attribute value of the link or that bit is irrelevant.
		  • If a bit in the mask is 1, the attribute value of that link and the required affinity of the LSP for that bit must match.

Step 5	**bandwidth** [**global**]*kbps* **Example:** `Device(config-lsp-attr)# bandwidth 5000`	(Optional) Specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295.
Step 6	**list** **Example:**	(Optional) Displays the contents of the LSP attribute list.

	Command or Action	Purpose
	`Device(config-lsp-attr)# list`	
Step 7	**lockdown** **Example:** `Device(config-lsp-attr)# lockdown`	(Optional) Disables reoptimization of the LSP.
Step 8	**priority** *setup-priority* [*hold-priority*] **Example:** `Device(config-lsp-attr)# priority 1 1`	(Optional) Specifies the LSP priority. • The *setup-priority* argument is used when signaling an LSP to determine which existing LSPs can be preempted. Valid values are from 0 through 7, where a lower number indicates a higher priority. Therefore, an LSP with a setup priority of 0 can preempt any LSP with a non-0 priority. • The *hold-priority* argument is associated with an LSP to determine if it should be preempted by other LSPs that are being signaled. Valid values are from 0 through 7, where a lower number indicates a higher priority.
Step 9	**record-route** **Example:** `Device(config-lsp-attr)# record-route`	(Optional) Records the route used by the LSP.
Step 10	**no** *sub-command* **Example:** `Device(config-lsp-attr)# no record-route`	(Optional) Removes a specific attribute from the LSP attributes list. • The *sub-command*argument names the LSP attribute to remove from the attributes list.
Step 11	**exit** **Example:** `Device(config-lsp-attr)# exit`	(Optional) Exits from LSP Attributes configuration mode.
Step 12	**end** **Example:** `Device(config)# end`	(Optional) Exits to privileged EXEC mode.

Adding Attributes to an LSP Attribute List

Perform this task to add attributes to an LSP attribute list. The LSP attribute list provides a user interface that is flexible, and easy to use. You can extend or change the LSP attribute list at any time to meet the requirements of your MPLS TE tunnel traffic. LSP Attributes configuration mode is used to display the specific LSP attributes list and to add or change the required path option attribute.

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 traffic-eng lsp attributes** *string* **Example:** `Device(config)# mpls traffic-eng lsp attributes 1`	Configures an LSP Attribute list and enters LSP Attributes configuration mode. • The *string* argument identifies a specific LSP Attribute list.

Step 4	**affinity** *value* [**mask***value*]
	Example:
	`Device(config-lsp-attr)# affinity 0 mask 0`
		(Optional) Specifies attribute flags for links comprising an LSP.
		• The *value* argument is a value required for links that make up an LSP. Values of the bits are either 0 or 1.
		• The **mask***value* keyword argument combination indicates which attribute values should be checked.
		  • If a bit in the mask is 0, an attribute value of the link or that bit is irrelevant.
		  • If a bit in the mask is 1, the attribute value of that link and the required affinity of the LSP for that bit must match.

Step 5	**bandwidth** [**global**] *kbps* **Example:** `Device(config-lsp-attr)# bandwidth 1000`	Specifies an LSP bandwidth. • The **global** keyword indicates a global pool path option.

	Command or Action	Purpose
		• The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295.
Step 6	**priority** *setup-priority* [*hold-priority*] **Example:** `Device(config-lsp-attr)# priority 2 2`	Specifies the LSP priority. • The *setup-priority* argument is used when signaling an LSP to determine which existing LSPs can be preempted. Valid values are from 0 through 7, where a lower number indicates a higher priority. Therefore, an LSP with a setup priority of 0 can preempt any LSP with a non-0 priority. • The *hold-priority* argument is associated with an LSP to determine if it should be preempted by other LSPs that are being signaled. Valid values are from 0 through 7, where a lower number indicates a higher priority.
Step 7	**list** **Example:** `Device(config-lsp-attr)# list`	(Optional) Displays the contents of the LSP attribute list. • Use the **list** command to display the path option attributes added to the attribute list.
Step 8	**exit** **Example:** `Device(config-lsp-attr)# exit`	(Optional) Exits LSP Attributes configuration mode.
Step 9	**end** **Example:** `Device(config)# end`	(Optional) Exits to privileged EXEC mode.

Example: Removing an Attribute from an LSP Attribute List

The following example shows how to remove the priority attribute from the LSP attribute list identified by the string 'simple'.

```
Device(config)# mpls traffic-eng lsp attributes simple
Device(config-lsp-attr)# priority 1 1
Device(config-lsp-attr)# list
LIST simple
 priority 1 1
```

```
!
Device(config-lsp-attr)# no priority
Device(config-lsp-attr)# list
LIST simple
!
Device(config-lsp-attr)# exit
```

Modifying an Attribute in an LSP Attribute List

Perform this task to modify an attribute in an LSP attribute list. The LSP attribute list provides a flexible user interface. You can extend or modifiy the LSP attribute list any time to meet the requirements of your MPLS TE tunnel traffic. LSP Attributes configuration mode is used to display the specific LSP attributes list and to modify the required path option attribute.

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 traffic-eng lsp attributes** *string* **Example:** `Device(config)# mpls traffic-eng lsp attributes 1`	Configures an LSP Attribute list and enters LSP Attributes configuration mode. • The *string* argument identifies a specific LSP attribute list.

Step 4	**affinity** *value* [**mask***value*]
	Example:
	`Device(config-lsp-attr)# affinity 1 mask 1`
		Specifies attribute flags for links comprising an LSP.
		• The *value* argument is a value required for links comprising an LSP. Values of bits are either 0 or 1.
		• The **mask***value* keyword argument combination indicates which attribute values should be checked.
		  • If a bit in the mask is 0, an attribute value of the link or that bit is irrelevant.
		  • If a bit in the mask is 1, the attribute value of that link and the required affinity of the tunnel for that bit must match.

	Command or Action	Purpose
Step 5	**list** **Example:** `Device(config-lsp-attr)# list`	(Optional) Displays the contents of the LSP Attribute list. • Use the **list** command to display the path option attributes configured in the attribute list.

Step 6	**affinity** *value* [**mask***value*]
	Example:
	`Device(config-lsp-attr)# affinity 0 mask 0`
		Specifies attribute flags for links comprising an LSP.
		• The *value* argument is a value required for links comprising an LSP. Values of bits are either 0 or 1.
		• The **mask***value* keyword argument combination indicates which attribute values should be checked.
		  • If a bit in the mask is 0, an attribute value of the link or that bit is irrelevant.
		  • If a bit in the mask is 1, the attribute value of that link and the required affinity of the tunnel for that bit must match.

Step 7	**list** **Example:** `Device(config-lsp-attr)# list`	(Optional) Displays the contents of the LSP attribute list. • Use the **list** command to verify that the path option attributes is modified in the attribute list.
Step 8	**exit** **Example:** `Device(config-lsp-attr)# exit`	(Optional) Exits LSP Attributes configuration mode.
Step 9	**end** **Example:** `Device(config)# end`	(Optional) Exits to privileged EXEC mode.

Deleting an LSP Attribute List

Perform this task to delete an LSP attribute list. You would perform this task when you no longer require the LSP attribute path options specified in the LSP attribute list for an MPLS TE 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	**no mpls traffic-eng lsp attributes** *string* **Example:** `Device(config)# no mpls traffic-eng lsp attributes 1`	Removes a specified LSP Attribute list from the device configuration. • The *string* argument identifies the specific LSP attribute list to remove.
Step 4	**end** **Example:** `Device(config)# end`	(Optional) Exits to privileged EXEC mode.
Step 5	**show mpls traffic-eng lsp attributes** [*string*] **Example:** `Device# show mpls traffic-eng lsp attributes`	(Optional) Displays information about configured LSP attribute lists. • Use the **show mpls traffic-eng lsp attributes** command to verify that the LSP attribute list was deleted from the Device.

Verifying Attributes Within an LSP Attribute List

To verify the attributes within an LSP attribute list, 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.

	Command or Action	Purpose
Step 3	**mpls traffic-eng lsp attributes** *string* **list** **Example:** `Device(config)# mpls traffic-eng lsp attributes 1 list`	Enters LSP Attributes configuration mode. Verifies the contents for a specific LSP attribute list.
Step 4	**exit** **Example:** `Device(config-lsp-attr)# exit`	Exits LSP Attributes configuration mode.
Step 5	**end** **Example:** `Device(config)# exit`	Exits to privileged EXEC mode.

Verifying All LSP Attribute Lists

Perform this task to verify all configured LSP attribute lists. Use this task to display all LSP attribute lists to verify that the attributes lists that you configured are in operation.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**show mpls traffic-eng lsp attributes** *string* [**details**] **Example:** `Device# show mpls traffic-eng lsp attributes`	Displays all configured LSP attribute lists.
Step 3	**show running-config** | **begin** *text-string* **Example:** `Device# show running-config | begin mpls traffic-eng lsp`	Verifies that all configured LSP attribute lists are as expected. The **begin** command modifier with the**mpls traffic-eng lsp** *text-string* locate the LSP attributes information in the configuration file.
Step 4	**exit** **Example:** `Device# exit`	Exits to user EXEC mode.

Associating an LSP Attribute List with a Path Option for an MPLS TE Tunnel

Perform this task to associate an LSP attribute list with a path option for an MPLS TE tunnel. This task is required if you want to apply the LSP attribute list that you configured to path options for your MPLS TE tunnels.

Based on your requirements, you can configure LSP attributes lists with different sets of attributes for different path options. LSP attribute lists also provide an easy way to configure multiple TE tunnels to use the same LSP attributes. That is, you can reference the same LSP attribute list to configure LSP-specific parameters for one or more TE tunnels.

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 tunnel 1`	Configures an interface type and enters interface configuration mode. • The *type* argument is the type of interface that you want to configure. • The *number* argument is the number of the tunnel interface that you want to create or configure.
Step 4	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.10.10.12`	Specifies the destination of the tunnel for this path option. • The *hostname* argument is the name of the host destination. • The *ip-address* argument is the IP address of the host destination expressed in decimal in four-part, dotted notation.
Step 5	**tunnel mode mpls traffic-eng** **Example:** `Device(config-if)# tunnel mode mpls traffic-eng`	Sets the encapsulation mode for the tunnel for MPLS TE.

	Command or Action	Purpose
Step 6	**tunnel mpls traffic-eng autoroute announce** **Example:** `Device(config-if)# tunnel mpls traffic-eng autoroute announce`	Specifies that the IGP should use the tunnel (if the tunnel is up) in its enhanced shortest path first (SPF) calculation.
Step 7	**tunnel mpls traffic-eng bandwidth** [**global**] *bandwidth* **Example:** `Device(config-if)# tunnel mpls traffic-eng bandwidth 1000`	Configures the bandwidth required for an MPLS TE tunnel and assigns it either to the subpool or the global pool. • The **global** keyword indicates a global pool tunnel. • The *kbps* argument is the bandwidth, in kilobits per second, set aside for the MPLS TE tunnel. The range is from 1 through 4294967295.
Step 8	**tunnel mpls traffic-eng priority** *setup-priority* [*hold-priority*] **Example:** `Device(config-if)# tunnel mpls traffic-eng priority 1 1`	Sets the priority to be used when the system determines which existing tunnels are eligible to be preempted. • The *setup-priority* argument is the priority used when signaling an LSP for this tunnel to determine which existing tunnels can be preempted. Valid values are from 0 through 7. A lower number indicates a higher priority. An LSP with a setup priority of 0 can preempt any LSP with a non-0 priority. • The *hold-priority* argument is the priority associated with an LSP for this tunnel to determine if it should be preempted by other LSPs that are being signaled. Valid values are from 0 through 7, where a lower number indicates a higher priority.
Step 9	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | *path-number*} [**verbatim**]} [**attributes** *string*] [**bandwidth** [**global**] *kbps*] [**lockdown**] **Example:** `Device(config-if)# tunnel mpls traffic-eng path-option 1 dynamic attributes 1`	Adds an LSP attribute list to specify LSP-related parameters for a path option for an MPLS TE tunnel. • The *number* argument identifies the path option. • The **dynamic** keyword indicates that the path option is dynamically calculated (the Device figures out the best path).

	Command or Action	Purpose
		• The **explicit** keyword indicates that the path option is specified. You specify the IP addresses of the path. • The **name** *path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option. • The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes** *string* keyword argument combination names an attribute list to specify path options for the LSP. • The **bandwidth** keyword specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 10	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.

Modifying a Path Option to Use a Different LSP Attribute List

Perform this task to modify the path option to use a different LSP Attribute list.

Based on your requirements, you can configure LSP attributes lists with different sets of attributes for different path options. You can change the set of attributes associated with a path option. The **tunnel mpls traffic-eng path-option** *number* **dynamic attributes** *string* command is used in interface configuration mode to modify the path option to use a different LSP attribute list. The **attributes** and *string* keyword and argument names the new LSP attribute list for the path option specified.

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 tunnel 1`	Configures the interface type and enters interface configuration mode. • The *type* argument is the type of interface that you want to configure. • The *number* argument is the number of the tunnel interface that you want to create or configure.
Step 4	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.10.10.12`	Specifies the destination of the tunnel for this path option. • The *hostname* argument is the name of the host destination. • The *ip-address* argument is the IP address of the host destination expressed in decimal in four-part, dotted notation.
Step 5	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name***path-name* | *path-number*} [**verbatim**]} [**attributes***string*] [**bandwidth** [**global**] *kbps*] [**lockdown**] **Example:** `Device(config-if)# tunnel mpls traffic-eng path-option 1 dynamic attributes 1`	Adds an LSP Attribute list to specify LSP-related parameters for a path option for an MPLS TE tunnel. • The *number* argument identifies the path option. • The **dynamic** keyword indicates that the path option is dynamically calculated (the Device figures out the best path). • The **explicit** keyword indicates that the path option is specified. You specify the IP addresses of the path. • The **name***path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option.

	Command or Action	Purpose
		• The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes***string* keyword argument combination names an attribute list to specify path options for the LSP. • The **bandwidth** keyword specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 6	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.

Removing a Path Option for an LSP for an MPLS TE Tunnel

Perform this task to remove a path option for an LSP for an MPLS TE tunnel. Use this task to remove a path option for an LSP when your MPLS TE tunnel traffic requirements change.

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	**interface** *type number* **Example:** `Device(config)# interface tunnel 1`	Configures the interface type and enters interface configuration mode. • The *type* argument is the type of interface that you want to configure. • The *number* argument is the number of the tunnel interface that you want to create or configure.
Step 4	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.10.10.12`	Specifies the destination of the tunnel for this path option. • The *hostname* argument is the name of the host destination. • The *ip-address* argument is the IP address of the host destination expressed in decimal in four-part, dotted notation.
Step 5	**no tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name***path-name* | *path-number*} [**verbatim**]} [**attributes***string*] [**bandwidth** [**global**] *kbps*] [**lockdown**] **Example:** `Device(config-if)# no tunnel mpls traffic-eng path-option 1 dynamic attributes 1`	Removes an LSP Attribute list that specifies LSP-related parameters for a path option for an MPLS TE tunnel. • The *number* argument identifies the path option. • The **dynamic** keyword indicates that the path option is dynamically calculated (the Device figures out the best path). • The **explicit** keyword indicates that the path option is specified. You specify the IP addresses of the path. • The **name***path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option. • The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes***string* keyword argument combination names an attribute list to specify path options for the LSP.

	Command or Action	Purpose
		• The **bandwidth** keyword specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 6	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.

Verifying that LSP Is Signaled Using the Correct Attributes

To verify that the LSP is signaled using the correct attributes for the specifed tunnel, 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	**show mpls traffic-eng tunnels** *tunnel-interface* [**brief**] **Example:** `Device# show mpls traffic-eng tunnels tunnel1`	Verifies that the LSP is signaled using the correct attributes for the specified tunnel.
Step 3	**exit** **Example:** `Device# exit`	Use this command to return to user EXEC mode. For example:

Configuring a Path Option for Bandwidth Override

The following section contains the tasks for configuring a path option for bandwidth override.

Note Once you configure bandwidth as a path-option parameter, you can no longer configure an LSP Attribute list as a path-option parameter.

Configuring Fallback Bandwidth Path Options for TE Tunnels

Perform this task to configure fallback bandwidth path options for a TE tunnel. Use this task to configure path options that reduce the bandwidth constraint each time the headend of a tunnel fails to establish an LSP.

Configuration of the Path Option for Bandwidth Override feature can reduce bandwidth constraints on path options temporarily. It improves the chances that an LSP is set up for the TE tunnel. When a TE tunnel uses a path option with bandwidth override, the traffic engineering software attempts every 30 seconds to reoptimize the tunnel to use the preferred path option with the original configured bandwidth. The Path Option for Bandwidth Override feature is designed as a temporary reduction in bandwidth constraint. To force immediate reoptimization of all traffic engineering tunnels, you can use the **mpls traffic-eng reoptimize** command. You can also configure the **lockdown** command with bandwidth override to prevent automatic reoptimization.

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 tunnel 1`	Configures an interface type and enters interface configuration mode. • The *type* argument is the type of interface that you want to configure. • The *number* argument is the number of the tunnel interface that you want to create or configure.
Step 4	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.10.10.12`	Specifies the destination of the tunnel for this path option. • The *hostname* argument is the name of the host destination. • The *ip-address* argument is the IP address of the host destination expressed in decimal in four-part, dotted notation.

	Command or Action	Purpose
Step 5	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | *path-number*} [**verbatim**]} [**attributes** *string*] [**bandwidth** [**global**] *kbps*] [**lockdown**] **Example:** `Device(config-if)# tunnel mpls traffic-eng path-option 1 dynamic bandwidth 500`	Adds a Path Option for Bandwidth Override to specify a bandwidth fallback for a path option for an MPLS TE tunnel. • The *number* argument identifies the path option. • The **dynamic** keyword indicates that the path option is dynamically calculated (the Device figures out the best path). • The **explicit** keyword indicates that the path option is specified. You specify the IP addresses of the path. • The **name***path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option. • The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes***string* keyword argument combination names an attribute list to specify path options for the LSP. • The **bandwidth** keyword specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 6	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.

Modifying the Bandwidth on a Path Option for Bandwidth Override

Perform this task to modify the bandwidth on a Path Option for Bandwidth Override. You might need to further reduce or modify the bandwidth constraint for a path option to ensure that the headend of a tunnel establishes an LSP.

The Path Option for Bandwidth Override feature is designed as a temporary reduction in bandwidth constraint. To force immediate reoptimization of all traffic engineering tunnels, you can use the **mpls traffic-eng reoptimize** command. You can also configure the **lockdown** command with bandwidth override to prevent automatic reoptimization.

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 tunnel 1`	Configures the interface type and enters interface configuration mode. • The *type* argument is the type of interface that you want to configure. • The *number* argument is the number of the tunnel interface that you want to create or configure.
Step 4	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.10.10.12`	Specifies the destination of the tunnel for this path option. • The *hostname* argument is the name of the host destination. • The *ip-address* argument is the IP address of the host destination expressed in decimal in four-part, dotted notation.
Step 5	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | *path-number*} [**verbatim**]} [**attributes** *string*] [**bandwidth** [**global**] *kbps*] [**lockdown**] **Example:** `Device(config-if)# tunnel mpls`	Adds a Path Option for Bandwidth Override to specify a bandwidth fallback for a path option for an MPLS TE tunnel. • The *number* argument identifies the path option.

	Command or Action	Purpose
	`traffic-eng path-option 2 dynamic bandwidth 500` **Example:**	• The **dynamic** keyword indicates that the path option is dynamically calculated (the Device figures out the best path). • The **explicit** keyword indicates that the path option is specified. You specify the IP addresses of the path. • The **name***path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option. • The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes***string* keyword argument combination names an attribute list to specify path options for the LSP. • The **bandwidth** keyword specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 6	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.
Step 7	**show mpls traffic-eng tunnels** *tunnel-interface* [**brief**] **Example:** `Device# show mpls traffic-eng tunnels tunnel1`	(Optional) Displays information about tunnels. • Use the **showmplstraffic-engtunnels** command to verify which bandwidth path option is in use by the LSP.

Removing a Path Option for Bandwidth Override

Perform this task to remove the bandwidth on the path option for bandwidth override. The Path Option for Bandwidth Override feature is designed as a temporary reduction in bandwidth constraint. Use this task to remove the bandwidth override when it is not required.

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** *number* **Example:** `Device(config)# interface tunnel 1`	Configures a tunnel interface type and enters interface configuration mode. • The *number* argument is the number of the tunnel interface that you want to create or configure.
Step 4	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.10.10.12`	Specifies the destination of the tunnel for this path option. • The *hostname* argument is the name of the host destination. • The *ip-address* argument is the IP address of the host destination expressed in decimal in four-part, dotted notation.
Step 5	**no tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | *path-number*} [**verbatim**]} [**attributes** *string*] [**bandwidth** [**global**] *kbps*] [**lockdown**] **Example:** `Device(config-if)# no tunnel mpls traffic-eng path-option 2 dynamic bandwidth 500`	Removes a path option for bandwidth override that specifies a bandwidth fallback for a path option for an MPLS TE tunnel. • The *number* argument identifies the path option. • The **dynamic** keyword indicates that the path option is dynamically calculated (the Device figures out the best path). • The **explicit** keyword indicates that the path option is specified. You specify the IP addresses of the path.

	Command or Action	Purpose
		• The **name** *path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option. • The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes** *string* keyword argument combination names an attribute list to specify path options for the LSP. • The **bandwidth** keyword specifies LSP bandwidth. • The **global** keyword indicates a global pool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 through 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 6	**end** **Example:** `Device(config-if)# end`	(Optional) Exits to privileged EXEC mode.
Step 7	**show mpls traffic-eng tunnels** *tunnel-interface* [**brief**] **Example:** `Device# show mpls traffic-eng tunnels tunnel1`	(Optional) Displays information about tunnels. • Use the **show mpls traffic-eng tunnels** command to verify which bandwidth path option is in use by the LSP.

Verifying that LSP Is Signaled Using the Correct Bandwidth

To verify that the LSP is signaled using the correct bandwidth, 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	**show mpls traffic-eng tunnels** *tunnel-interface* [**brief**] **Example:** `Device# show mpls traffic-eng tunnels tunnel21`	Verifies that the LSP is signaled with the correct bandwidth. Overrides the bandwidth configured on the tunnel. If bandwidth override is actively being signaled, the **show mpls traffic-eng tunnel** command displays the bandwidth override information under the Active Path Option Parameters heading. The example shows that BandwidthOverride is enabled and that the tunnel is signaled using path-option 2. The bandwidth signaled is 500. This is the value configured on the path option 2 and it overrides the 1000 kbps bandwidth configured on the tunnel interface.
Step 3	**exit** **Example:** `Device# exit`	Use this command to exit to user EXEC mode. For example:

Configuration Examples for MPLS Traffic Engineering—LSP Attributes

The following section provides configuration examples for configuring MPLS Traffic Engineering—LSP Attributes.

Configuring LSP Attribute List Examples

Example: Configuring an LSP Attribute List

This example shows the configuration of the affinity and bandwidth LSP-related attributes in an LSP attribute list identified with the numeral 1.

```
Device(config)# mpls traffic-eng lsp attributes 1
Device(config-lsp-attr)# affinity 7 mask 7
Device(config-lsp-attr)# bandwidth 1000
Device(config-lsp-attr)# exit
```

Example: Adding Attributes to an LSP Attribute List

This example shows the addition of priority attributes to the LSP attribute list identified with the numeral 1.

```
Device(config)# mpls traffic-eng lsp attributes 1
Device(config-lsp-attr)# affinity 7 mask 7
Device(config-lsp-attr)# bandwidth 1000
Device(config-lsp-attr)# priority 1 1
Device(config-lsp-attr)# exit
```

Example: Removing an Attribute from an LSP Attribute List

The following example shows how to remove the priority attribute from the LSP attribute list identified by the string 'simple'.

```
Device(config)# mpls traffic-eng lsp attributes simple
Device(config-lsp-attr)# priority 1 1
Device(config-lsp-attr)# list
LIST simple
 priority 1 1
!
Device(config-lsp-attr)# no priority
Device(config-lsp-attr)# list
LIST simple
!
Device(config-lsp-attr)# exit
```

Example: Modifying an Attribute in an LSP Attribute List

The following example shows how to modify the bandwidth in an LSP attribute list identified by the numeral 5.

```
Device(config)# mpls traffic-eng lsp attributes 5
Device(config-lsp-attr)# bandwidth 1000
Device(config-lsp-attr)# priority 1 1
Device(config-lsp-attr)# list
LIST 5
 bandwidth 1000
 priority 1 1
Device(config-lsp-attr)# bandwidth 500
Device(config-lsp-attr)# list
LIST 5
 bandwidth 500
 priority 1 1
Device(config-lsp-attr)# exit
```

Example: Deleting an LSP Attribute List

The following example shows how to delete an LSP attribute list identified by the numeral 1.

```
Device(config)# mpls traffic-eng lsp attributes 1
Device(config-lsp-attr)# affinity 7 mask 7
Device(config-lsp-attr)# bandwidth 1000
Device(config-lsp-attr)# priority 1 1

Device(config-lsp-attr)# exit
```

```
!
Device(config)# no mpls traffic-eng lsp attributes 1
```

Example: Associating an LSP Attribute List with a Path Option for a TE Tunnel: Example

The following example associates the LSP attribute list identified by the numeral 3 with path option 1.

```
Device(config)# mpls traffic-eng lsp attributes 3
Device(config-lsp-attr)# bandwidth 1000
Device(config-lsp-attr)# priority 2 2
Device(config-lsp-attr)# exit
!
!
Device(config)# interface Tunnel 1
Device(config-if)# ip unnumbered FastEthernet1/0/1
Device(config-if)# tunnel destination 10.112.0.12
Device(config-if)# tunnel mode mpls traffic-eng
Device(config-if)# tunnel mpls traffic-eng affinity 1
Device(config-if)# tunnel mpls traffic-eng bandwidth 5000
Device(config-if)# tunnel mpls traffic-eng path-option 1 dynamic attributes 3
```

In this configuration, the LSP has the following attributes:

```
{bandwidth = 1000
 priority = 2 2
 affinity 1
 reroute enabled.
 }
```

The LSP attribute list referenced by the path option takes precedence over the values configured on the tunnel interface.

Example: Modifying a Path Option to Use a Different LSP Attribute List

The following example modifies path option 1 to use an LSP attribute list identified by the numeral 1.

```
Device(config)# mpls traffic-eng lsp attributes 1
Device(config-lsp-attr)# affinity 7 mask 7
Device(config-lsp-attr)# bandwidth 500
Device(config-lsp-attr)# priority 1 1
Device(config-lsp-attr)# exit
Device(config)# mpls traffic-eng lsp attributes 2
Device(config-lsp-attr)# bandwidth 1000
Device(config-lsp-attr)# priority 1 1
Device(config-lsp-attr)# exit
Device(config)# interface Tunnel 1
Device(config-if)# ip unnumbered FastEthernet1/0/1
Device(config-if)# tunnel destination 10.112.0.12
Device(config-if)# tunnel mode mpls traffic-eng
Device(config-if)# tunnel mpls traffic-eng affinity 1
Device(config-if)# tunnel mpls traffic-eng bandwidth 5000
Device(config-if)# tunnel mpls traffic-eng path-option 1 dynamic attributes 1
```

In this configuration, the LSP has the following attributes:

```
{affinity = 7 mask = 7
 bandwidth = 500
 priority = 1 1
 }
```

Example: Removing a Path Option for an LSP for an MPLS TE Tunnel

The following example shows the removal of path option 1 for an LSP for a TE tunnel.

```
Device(config)# interface Tunnel 1
Device(config-if)# ip unnumbered FastEthernet1/0/1
Device(config-if)# tunnel destination 10.112.0.12
Device(config-if)# tunnel mode mpls traffic-eng
Device(config-if)# tunnel mpls traffic-eng affinity 1
Device(config-if)# tunnel mpls traffic-eng bandwidth 5000
Device(config-if)# tunnel mpls traffic-eng path-option 1 explicit path1 attributes 1
Device(config-if)# tunnel mpls traffic-eng path-option 2 explicit path2 attributes 2
!
!
Device(config-if)# no tunnel mpls traffic-eng path-option 1 explicit path1 attributes 1
```

Configuring a Path Option for Bandwidth Override Examples

Example: Configuring a Path Option to Override the Bandwidth

The following examples show how to configure a path option to override the bandwidth:

```
Device(config-if)# tunnel mpls traffic-eng path-option 3 explicit name path1 ?
       attributes Specify an LSP attribute list
       bandwidth  override the bandwidth configured on the tunnel
       lockdown   not a candidate for reoptimization
       <cr>
Device(config-if)# tunnel mpls traffic-eng path-option 3 explicit name path1 bandwidth ?
       <0-4294967295>  bandwidth requirement in kbps
Device(config-if)# tunnel mpls traffic-eng path-option 3 explicit name path1 bandwidth 500
 ?
       lockdown   not a candidate for reoptimization
       <cr>
```

Note Once you configure bandwidth as a path-option parameter, you can no longer configure an LSP attribute list as a path-option parameter.

Configuring Fallback Bandwidth Path Options for TE Tunnels: Example

The following example shows multiple path options configured with the **tunnel mpls traffic-eng path-option** command:

```
interface Tunnel 1
 ip unnumbered Loopback0
 tunnel destination 10.10.10.12
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name path1
 tunnel mpls traffic-eng path-option 2 explicit name path2 bandwidth 500
 tunnel mpls traffic-eng path-option 3 dynamic bandwidth 0
end
```

The device selects a path option for an LSP in order of preference, as follows:

- The device attempts to signal an LSP using path options starting with path-option 1.

The device attempts to signal an LSP with the 1000 kbps bandwidth configured on the tunnel interface because path-option 1 has no bandwidth configured.

- If 1000 kbps bandwidth is not available over the network, the device attempts to establish an LSP using path-option 2.

Path-option 2 has a bandwidth of 500 kbps configured. This reduces the bandwidth constraint from the original 1000 kbps configured on the tunnel interface.

- If 500 kbps is not available, the device attempts to establish an LSP using path-option 3.

Path-option 3 is configured as dynamic and has bandwidth 0. The device establishes the LSP if an IP path exists to the destination and all other tunnel constraints are met.

Example: Modifying the Bandwidth on a Path Option for Bandwidth Override

The following example shows modifying the bandwidth on a path option for bandwidth override. Path-option 3 is changed to an explicit path with a bandwidth of 100 kbps. Path-option 4 is configured with bandwidth 0.

```
interface Tunnel 1
 ip unnumbered Loopback0
 tunnel destination 10.10.10.12
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name path1
 tunnel mpls traffic-eng path-option 2 explicit name path2 bandwidth 500
 tunnel mpls traffic-eng path-option 3 dynamic bandwidth 0
!
!
Device(config)# tunnel mpls traffic-eng path-option 3 explicit name path3 bandwidth 100
Device(config)# tunnel mpls traffic-eng path-option 4 dynamic bandwidth 0
```

Example:Removing the Path Option Bandwidth Value for an LSP for an MPLS TE Tunnel

The following example shows the removal of the bandwidth for path option 3 for an LSP for an MPLS TE tunnel:

```
interface Tunnel 1
 ip unnumbered Loopback0
 tunnel destination 10.10.10.12
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng path-option 1 explicit name path1
 tunnel mpls traffic-eng path-option 2 explicit name path2 bandwidth 500
 tunnel mpls traffic-eng path-option 3 explicit name path3 bandwidth 100
 tunnel mpls traffic-eng path-option 4 dynamic bandwidth 0
!
Router(config)# no tunnel mpls traffic-eng path-option 3 explicit name path3 bandwidth 100
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
MPLS traffic engineering commands	Cisco IOS Multiprotocol Label Switching Command Reference

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 Traffic Engineering—LSP Attributes

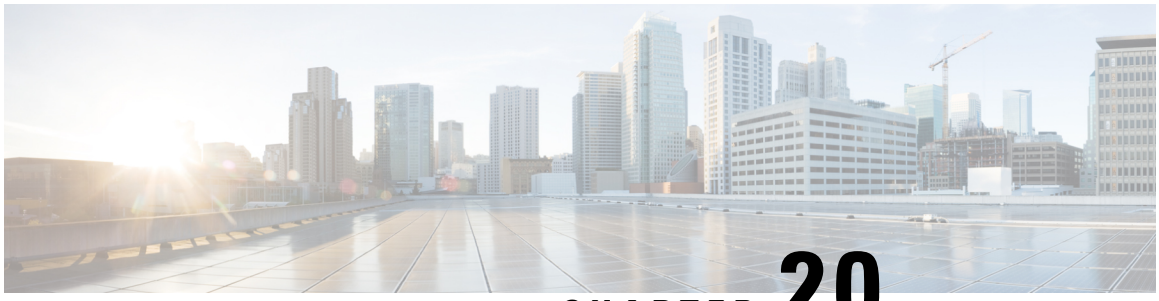
This table provides release and related information for the features explained in this module.

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 Bengaluru 17.6.1	MPLS Traffic Engineering LSP Attributes	The MPLS Traffic Engineering—LSP Attributes feature is an extension to MPLS TE that provides an LSP Attribute List feature and a Path Option for Bandwidth Override feature. These features provide flexibility in the configuration of LSP attributes for MPLS TE tunnel path options. Several LSP attributes can be applied to path options for TE tunnels using an LSP attribute list. If bandwidth is the only LSP attribute you require, then you can configure a Path Option for Bandwidth Override.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	MPLS Traffic Engineering LSP Attributes	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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/



CHAPTER 20

Configuring MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

Prerequisites for MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

Before you configure tunnel path calculation metrics, your network must support the following Cisco IOS XE features:

- Multiprotocol Label Switching (MPLS) traffic engineering tunnels
- IP Cisco Express Forwarding
- Open Shortest Path First (OSPF) or Intermediate System-to-Intermediate System (IS-IS)

Restrictions for MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

- Unless explicitly configured, the TE link metric for a given link is the IGP link metric. When the TE link metric is used to represent a link property that is different from cost/distance, you must configure every network link that can be used for TE tunnels with a TE link metric that represents that property. You can do this by using the **mpls traffic-eng administrative-weight** command. Failure to do so might cause tunnels to use unexpected paths.
- MPLS traffic engineering supports only a single IGP process/instance. Multiple IGP processes/instances are not supported. MPLS traffic engineering should not be configured in more than one IGP process/instance.

Information About MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

The following section provides information about Configurable Path Calculation Metric for MPLS Traffic Engineering tunnels.

Overview

The MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels feature enables you to control the metric used in path calculation for traffic engineering (TE) tunnels on a per-tunnel basis.

When MPLS TE is configured in a network, the Interior Gateway Protocol (IGP) floods two metrics for every link. The metrics are the normal IGP (OSPF or IS-IS) link metric and a TE link metric. The IGP uses the IGP link metric in the normal way to compute routes for destination networks.

You can specify that the path calculation for a given tunnel is based on either of the following:

- IGP link metrics
- TE link metrics, which you can configure so that they represent the needs of a particular application. For example, the TE link metrics can be configured to represent link transmission delay.

Benefits

When Traffic Engineering (TE) tunnels carry two types of traffic, the Configurable Path Calculation Metric for Tunnels feature allows you to tailor tunnel path selection to the requirements of each type of traffic.

For example, suppose that certain tunnels are to carry voice traffic (which requires low delay) and other tunnels are to carry data. In this situation, you can use the TE link metric to represent link delay and do the following:

- Configure tunnels that carry voice to use the TE link metric set to represent link delay for path calculation.
- Configure tunnels that carry data to use the IGP metric for path calculation.

How to Configure MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

The following section provides information about the configuration steps for Configurable Path Calculation Metric for Tunnels for MPLS Traffic Engineering.

Configuring a Platform to Support Traffic Engineering Tunnels

To configure a platform to support Traffic Engineering tunnels, 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	**ip cef distributed** **Example:** `Device(config)# ip cef distributed`	Enables distributed Cisco Express Forwarding operation.
Step 4	**mpls traffic-eng tunnels** **Example:** `Device(config)# mpls traffic-eng tunnels`	Enables the MPLS traffic engineering tunnel feature on a device.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring IS-IS for MPLS Traffic Engineering

Note MPLS traffic engineering supports only a single IGP process or instance. Multiple IGP processes or instances are not supported. MPLS traffic engineering should not be configured in more than one IGP process or instance.

To configure IS-IS for MPLS traffic engineering, perform 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 isis** **Example:** `Device(config)# router isis`	Enables IS-IS routing and specifies an IS-IS process. The device enters configuration mode.
Step 4	**mpls traffic-eng level** **Example:** `Device(config-router)# mpls traffic-eng level-1`	Turns on MPLS traffic engineering for IS-IS level 1.
Step 5	**mpls traffic-eng level** **Example:** `Device(config-router)# mpls traffic-eng level-2`	Turns on MPLS traffic engineering for IS-IS level 2.
Step 6	**mpls traffic-eng router-id** *type number* **Example:** `Device(config-router)# mpls traffic-eng router-id loopback 0`	Specifies that the traffic engineering router identifier for the node is the IP address associated with interface loopback0.
Step 7	**metric-style wide** **Example:** `Device(config-router)# metric-style wide`	Configures a router to generate and accept only new-style type, length, value objects (TLVs).

Configuring Traffic Engineering Link Metrics

Unless explicitly configured, the TE link metric is the IGP link metric.

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* [**.** *subinterface-number*] **Example:** `Device(config)# interface port channel 20`	Configures an interface type and enters interface configuration mode. • The *type* argument is the type of interface to be configured. • The *slot* argument is the chassis slot number. • The / *subslot* keyword and argument pair is the secondary slot number. The slash (/) is required. • The / *port* keyword and argument pair is the port or interface number. The slash (/) is required. • The **.** *subinterface-number*keyword and argument pair is the subinterface number in the range 1–4294967293. The number that precedes the period (.) must match the number to which this subinterface belongs.
Step 4	**mpls traffic-eng administrative-weight** *weight* **Example:** `Device(config-if)# mpls traffic-eng administrative-weight 20`	Overrides the IGP administrative weight (cost) of the link. • The *weight* argument is the cost of the link.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**exit** **Example:**	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)# exit`	

Configuring an MPLS Traffic Engineering Tunnel

To configure a preferred explicit path for an MPLS TE tunnel, 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 tunnel** *number* **Example:** `Device(config)# interface Tunnel0`	Configures an interface type and enters interface configuration mode. • The *number* argument is the number of the tunnel.
Step 4	**ip unnumbered** *type number* **Example:** `Device(config-if)# ip unnumbered loopback0`	Enables IP processing on an interface without assigning an explicit IP address to the interface. • The *type* and *number* arguments name the type and number of another interface on which the router has an assigned IP address. It cannot be another unnumbered interface. • An MPLS traffic engineering tunnel interface should be unnumbered because it represents a unidirectional link.
Step 5	**tunnel destination** *ip-address* **Example:** `Device(config-if)# tunnel destination 192.168.4.4`	Specifies the destination for a tunnel interface. • The *ip-address* argument must be the MPLS traffic engineering router ID of the destination device.
Step 6	**tunnel mode mpls traffic-eng** **Example:**	Sets the tunnel encapsulation mode to MPLS traffic engineering.

	Command or Action	Purpose
	`Device(config-if)# tunnel mode mpls traffic-eng`	
Step 7	**tunnel mpls traffic-eng bandwidth** *bandwidth* **Example:** `Device(config-if)# tunnel mpls traffic-eng bandwidth 250`	Configures the bandwidth for the MPLS traffic engineering tunnel. • The *bandwidth* argument is a number in kilobits per second that is set aside for the MPLS traffic engineering tunnel. Range is from 1 through 4294967295. **Note** If automatic bandwidth is configured for the tunnel, use the **tunnel mpls traffic-eng bandwidth** command to configure the initial tunnel bandwidth.
Step 8	**tunnel mpls traffic-eng path-option** *number* {**dynamic** | **explicit** {**name** *path-name* | **identifier** *path-number*}} [**lockdown**] **Example:** `Device(config-if)# tunnel mpls traffic-eng path-option 10 explicit identifier 321`	Configures the tunnel to use a named IP explicit path or a path dynamically calculated from the traffic engineering topology database. • The *number* argument is the preference for this path option. When you configure multiple path options, lower numbered options are preferred. Valid values are from 1 through 1000. • The **dynamic** keyword indicates that the path of the LSP is dynamically calculated. • The **explicit** keyword indicates that the path of the LSP is an IP explicit path. • The **name** *path-name* keyword and argument are the path name of the IP explicit path that the tunnel uses with this option. • The **identifier** *path-number* keyword and argument pair names the path number of the IP explicit path that the tunnel uses with this option. The range is from 1 through 65535. • The **lockdown** keyword specifies that the LSP cannot be reoptimized. **Note** A dynamic path is used if an explicit path is currently unavailable.

	Command or Action	Purpose
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.

Configuring the Metric Type for Tunnel Path Calculation

Unless explicitly configured, the traffic engineering link metric type is used for tunnel path calculation. You can use two commands to control the metric type to use: an interface configuration command that specifies the metric type to be used for a particular TE tunnel. And a global configuration command that specifies the metric type to use for TE tunnels for which a metric type is unspecified by the interface configuration command.

Note If you do not enter either of the path selection metrics commands, the traffic engineering (TE) metric is used.

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** *number* **Example:** `Device(config)# interface Tunnel0`	Configures an interface type and enters interface configuration mode. • The *number* argument is the number of the tunnel.
Step 4	**tunnel mpls traffic-eng path-selection metric** {**igp** | **te**} **Example:** `Device(config-if)# tunnel mpls traffic-eng path-selection metric igp`	Specifies the metric type to use for path calculation for a tunnel. • The **igp** keyword specifies the use of the Interior Gateway Protocol (IGP) metric. • The **te** keyword specifies the use of the traffic engineering (TE) metric. The traffic engineering metric is the default.
Step 5	**exit** **Example:**	Exits interface configuration mode and returns to global configuration mode.

	Command or Action	Purpose
	`Device(config-if)# exit`	
Step 6	**mpls traffic-eng path-selection metric** {**igp** | **te**} **Example:** `Device(config)# mpls traffic-eng path-selection metric igp`	Specifies the metric type to use when a metric type was not explicitly configured for a given tunnel. • The **igp** keyword specifies the use of the Interior Gateway Protocol (IGP) metric. • The **te** keyword specifies the use of the traffic engineering (TE) metric. This is the default.
Step 7	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Verifying the Tunnel Path Metric Configuration

To verify the tunnel path metric configuration, perform the following procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**show mpls traffic-eng topolog** y **Example:** `Device# show mpls traffic-eng topology`	Displays TE and IGP metrics for each link. You can verify that link metrics are correctly configured for a network.
Step 3	**show mpls traffic-eng tunnels** **Example:** `Device# show mpls traffic-eng tunnels`	Displays the link metric used for tunnel path calculation. You can verify that the desired link metrics are used for each tunnel.
Step 4	**exit** **Example:** `Device# exit`	Returns to user EXEC mode.

Configuration Examples for MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

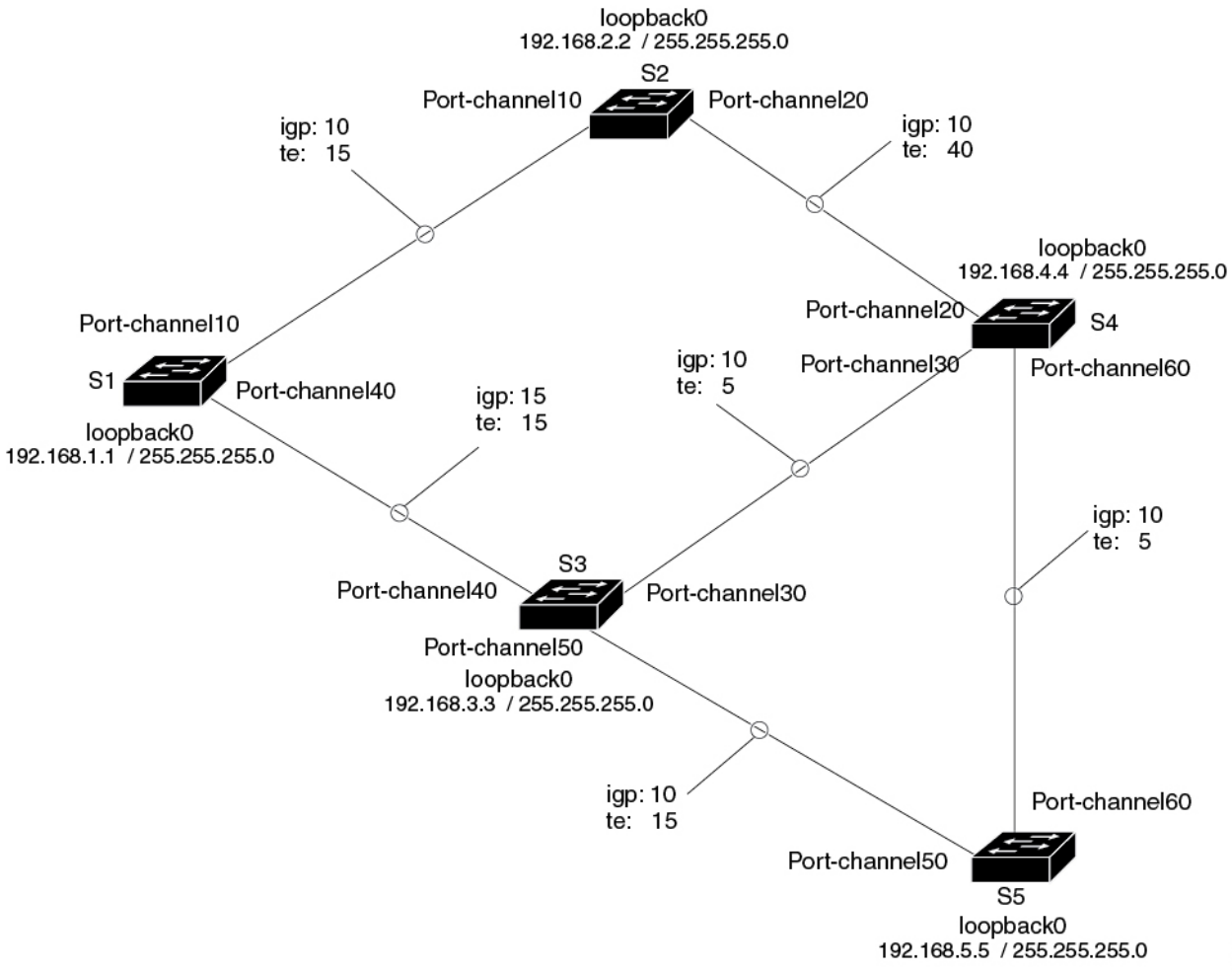
The following section provides configuration examples for configuring a path calculation metric for tunnels.

Example: Configuring Link Type and Metrics for Tunnel Path Selection

The section illustrates how to configure the link metric type to use for tunnel path selection. And how to configure the link metrics themselves. The configuration commands included focus on specifying the metric type for path calculation and assigning metrics to links. You will need additional commands are required to fully configure the example scenario. For example, the IGP commands for traffic engineering and the link interface commands for enabling traffic engineering and specifying available bandwidth.

The examples in this section support the simple network technology shown in the following figure.

Figure 24: Network topology



In the figure above:

- Tunnel1 and Tunnel2 run from S1 (headend) to S4 (tailend).
- Tunnel3 runs from S1 to S5.
- Path calculation for Tunnel1 and Tunnel3 should use a metric that represents link delay because these tunnels carry voice traffic.
- Path calculation for Tunnel2 should use IGP metrics because MPLS TE carries data traffic with no delay requirement.

Configuration fragments follow for each of the devices that illustrate the configuration relating to link metrics and their use in tunnel path calculation. TE metrics that represent link delay must be configured for the network links on each of the devices. And the three tunnels must be configured on S1.

This configuration fragments force Tunnel1 to take path S1-S3-S4, Tunnel2 to take path S1-S2-S4, and Tunnel3 to take path S1-S3-S4-S5 (assuming the links have sufficient bandwidth to accommodate the tunnels).

S1 Configuration

The following example shows how to configure the tunnel headend S1 for Tunnel1, Tunnel2, and Tunnel3 in the preceding figure:

```
interface port channel 40
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channel 10
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface Tunnel1                               !Tunnel1 uses TE metric (default)
                                                !for path selection
ip unnumbered loopback0
tunnel destination 192.168.4.4 255.255.255.0
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng bandwidth 1000
tunnel mpls traffic-eng path-option 1 dynamic
interface Tunnel2                                !Tunnel2 uses IGP metric
                                                 !for path selection
ip unnumbered loopback0
tunnel destination 192.168.4.4 255.255.255.0
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng bandwidth 1000
tunnel mpls traffic-eng path-option 1 dynamic
tunnel mpls traffic-eng path-selection-metric igp !Use IGP cost for path selection.
interface Tunnel3                                  !Tunnel3 uses TE metric (default)
                                                   !for path selection
ip unnumbered loopback0
tunnel destination 192.168.5.5 255.255.255.0
tunnel mode mpls traffic-eng
tunnel mpls traffic-eng bandwidth 1000
tunnel mpls traffic-eng path-option 1 dynamic
```

S2 Configuration

The following example shows how to configure S2 in the preceding figure:

```
interface port channel 10
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channle 20
mpls traffic-eng administrative-weight 40       !TE metric different from IGP metric
```

S3 Configuration

The following example shows how to configure S3 in the preceding figure:

```
interface port channel 40
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channel 50
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channel 30
mpls traffic-eng administrative-weight 5        !TE metric different from IGP metric
```

S4 Configuration

The following example shows how to configure R4 in the preceding figure:

```
interface port channel 20
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channel 30
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channel 60
mpls traffic-eng administrative-weight 5        !TE metric different from IGP metric
```

S5 Configuration

The following example shows how to configure S5 in the preceding figure:

```
interface port channel 50
mpls traffic-eng administrative-weight 15       !TE metric different from IGP metric
interface port channel 60
mpls traffic-eng administrative-weight 5        !TE metric different from IGP metric
```

Example: Verifying the Tunnel Path Metric Configuration

The following examples show how to verify the Tunnel Path Metric configuration.

The following example is a sample output of the **show mpls traffic-eng topology** command. This command displays the TE and IGP metrics for each link.

```
Device# show mpls traffic-eng topology
My_System_id: 1440.0000.0044.00 (isis  level-1)
IGP Id: 0090.0000.0009.00, MPLS TE Id:192.168.9.9 Router Node  (isis  level-1)
      link[0 ]:Nbr IGP Id: 0090.0000.0009.03, gen:7
          frag_id 0, Intf Address:10.0.0.99
          TE metric:100, IGP metric:48, attribute_flags:0x0     !!Note TE and IGP metrics
          physical_bw: 10000 (kbps), max_reservable_bw_global: 0 (kbps)
          max_reservable_bw_sub: 0 (kbps)
.
.
.
      link[1 ]:Nbr IGP Id: 0055.0000.0055.00, gen:7
          frag_id 0, Intf Address:10.205.0.9, Nbr Intf Address:10.205.0.55
          TE metric:120, IGP metric:10, attribute_flags:0x0     !!Note TE and IGP metrics
          physical_bw: 155000 (kbps), max_reservable_bw_global: 500000 (kbps)
          max_reservable_bw_sub: 0 (kbps)
```

The following example is a sample output of the **show mpls traffic-eng tunnels** command. This command displays the link metric used for tunnel path calculation.

```
Device# show mpls traffic-eng tunnels
Name: te3640-17-c_t221              (Tunnel22) Destination: 192.168.100.22
  Status:
    Admin: up         Oper: up     Path: valid       Signalling: connected
    path option 1, type dynamic (Basis for Setup, path weight 10)
  Config Parameters:
    Bandwidth: 400 kps (Global)    Priority: 1  1   Affinity: 0x0/0xFFFF
    Metric Type: IGP                                        !!Note metric type
    AutoRoute:  enabled   LockDown: disabled  Loadshare: 0   bw-based
    auto-bw: disabled(0/115) 0  Bandwidth Requested: 0
.
.
.
Name: te3640-17-c_t222              (Tunnel33) Destination: 192.168.100.22
  Status:
    Admin: up         Oper: up     Path: valid       Signalling: connected
    path option 1, type dynamic (Basis for Setup, path weight 10)
  Config Parameters:
    Bandwidth: 200 kbps (Global)   Priority: 1  1   Affinity: 0x0/0xFFFF
    Metric Type: TE                                          !!Note metric type
    AutoRoute:  enabled   LockDown: disabled  Loadshare: 0   bw-based
    auto-bw: disabled(0/115) 0  Bandwidth Requested: 0
.
.
.
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
Configuration tasks for IS-IS and OSPF	*Cisco IOS XE IP Routing Protocols Configuration Guide*
IS-IS and OSPF commands	*Cisco IOS IP Routing Protocols Command Reference*
Configuration tasks for MPLS and MPLS TE	*Cisco IOS XE Multiprotocol Label Switching Configuration Guide*
MPLS TE commands	*Cisco IOS Multiprotocol Label Switching Command Reference*
Configuration tasks for tunnels	• *Cisco IOS XE Interface and Hardware Component Configuration Guide* • *Cisco IOS XE Multiprotocol Label Switching Configuration Guide*
Tunnel configuration commands	• *Cisco IOS Interface and Hardware Component Command Reference* • *Cisco IOS XE Multiprotocol Label Switching Command Reference*

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

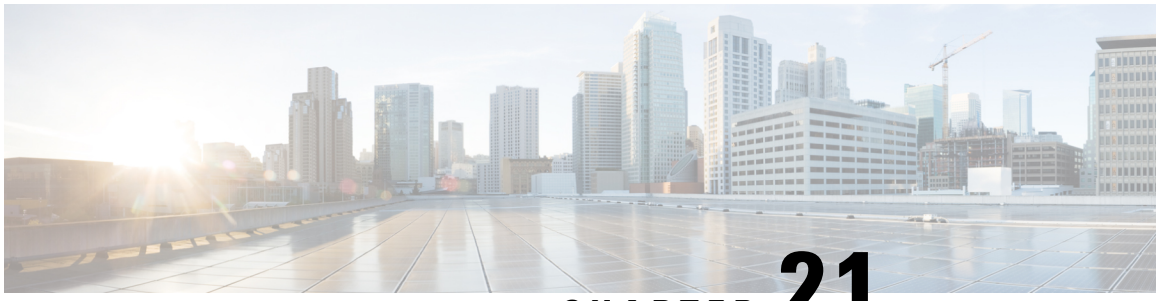
Feature History for MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels

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 Bengaluru 17.6.1	MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels	The MPLS Traffic Engineering—Configurable Path Calculation Metric for Tunnels feature enables you to control the metric used in path calculation for traffic engineering (TE) tunnels on a per-tunnel basis.

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/



CHAPTER 21

Configuring MPLS Traffic Engineering—RSVP Graceful Restart

Prerequisites for MPLS TE—RSVP Graceful Restart

Perform the following tasks on devices before configuring the MPLS Traffic Engineering—RSVP Graceful Restart feature:

- Configure the Resource Reservation Protocol (RSVP).
- Enable MPLS.
- Configure traffic engineering (TE).
- Enable graceful restart.

Restrictions for MPLS TE—RSVP Graceful Restart

- Graceful restart supports node failure only.
- Cisco recommends that you configure interface hellos only if the neighbor device does not support node hellos.
- Unnumbered interfaces are not supported.
- You cannot configure an interface hello for graceful restart and a hello state timeout (HST) on the same interface.

Information About MPLS TE—RSVP Graceful Restart

The following section provides information about MPLS TE—RSVP Graceful Restart.

Graceful Restart Operation

The MPLS Traffic Engineering—RSVP Graceful Restart feature allows a neighboring Route Processor (RP) to recover from disruption in control plane service (specifically, the Label Distribution Protocol (LDP) component) without losing its Multiprotocol Label Switching (MPLS) forwarding state. This feature has the following benefits:

- Graceful restart allows a node to recover state information from its neighbor when there is an RP failure or the device has undergone a stateful switchover (SSO).
- Graceful restart allows session information recovery with minimal disruption to the network.
- A node can perform a graceful restart to help a neighbor recover its state by keeping the label bindings and state information to provide a quick recovery of the failed node and not affect the traffic that is currently forwarded.

The node failure may be completely transparent to other nodes in the network.

RSVP graceful restart preserves the label values and forwarding information and works with third-party or Cisco Devices seamlessly.

RSVP graceful restart depends on RSVP hello messages to detect that a neighbor went down. Hello messages include Hello Request or Hello Acknowledgment (ACK) objects between two neighbors.

A node hello is transmitted when graceful restart is globally configured and the first LSP to the neighbor is created.

Interface hello is an optional configuration. If you configure the graceful restart Hello command on an interface, the interface hello is considered to be an additional hello instance with the neighbor.

The Device transmits an interface hello for graceful restart when all of the following conditions are met:

- Graceful restart is configured globally.
- Graceful restart is configured on the interface.
- An LSP to the neighboring Device is created and goes over the interface.

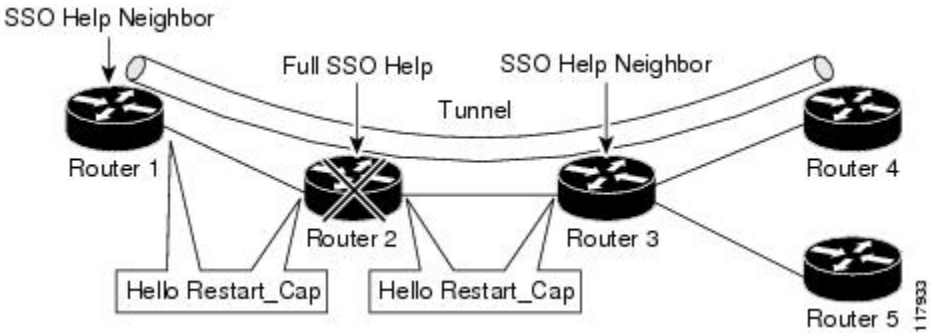
Cisco recommends that you use node hellos if the neighbor supports node hellos, and configure interface hellos only if the neighbor Device does not support node hellos.

Interface hellos differ from node hellos. as follows:

- **Interface hello** —The source address in the IP header of the hello message has an IP address that matches the interface that the Hello message sent out. The destination address in the IP header is the interface address of the neighbor on the other side of the link. A TTL of 1 is used for per-interface hellos as it is destined for the directly-connected neighbor.
- **Node hello** —The source address in the IP header of the Hello message includes the TE Device ID of the sending Device. The destination address of the IP header has the Device ID of the neighbor to which this message is sent. A TTL of more than 1 is used.

The figure below shows the graceful restart extension to these messages that an object called Restart_Cap, which tells neighbors that a node, may be capable of restarting if a failure occurs. The time-to-live (TTL) in these messages is set to 255 so that adjacencies can be maintained through alternate paths even if the link between two neighbors goes down.

Figure 25: How Graceful Restart Works



The Restart_Cap object has two values—the restart time, which is the sender's time to restart the RSVP_TE component and exchange hello messages after a failure; and the recovery time, which is the desired time that the sender wants the receiver to synchronize the RSVP and MPLS databases.

In the figure above, graceful restart is enabled on Device 1, Device 2, Device 3, and Device 4. For simplicity, assume that all Devices are restart capable. A TE label switched path (LSP) is signaled from Device 1 to Device 4.

Device 2 and Device 3 exchange periodic graceful restart hello messages every 10000 ms (10 seconds), and so do Device 2 and Device 1 and Device 3 and Device 4. Assume that Device 2 advertises its restart time as 60000 ms (60 seconds) and its recovery time as 60000 ms (60 seconds) as shown in the following example:

```
23:33:36: Outgoing Hello:
23:33:36:   version:1 flags:0000 cksum:883C ttl:255 reserved:0 length:32
23:33:36:   HELLO                type HELLO REQUEST length 12:
23:33:36:   Src_Instance: 0x6EDA8BD7, Dst_Instance: 0x00000000
23:33:36:   RESTART_CAP          type 1 length 12:
23:33:36:   Restart_Time: 0x0000EA60
, Recovery_Time: 0x0000EA60
```

Note The restart and recovery time are shown in **bold** in the last entry.

Device 3 records this into its database. Also, both neighbors maintain the neighbor status as UP. However, Device 3's control plane fails at some point (for example, a Primary Route Processor failure). As a result, RSVP and TE lose their signaling information and states although data packets continue to be forwarded by the line cards.

When four ACK messages are missed from Device 2 (40 seconds), Device 3 declares communication with Device 2 lost "indicated by LOST" and starts the restart time to wait for the duration advertised in Device 2's restart time previously and recorded (60 seconds). Device 1 and Device 2 suppress all RSVP messages to Device 3 except hellos. Device 3 keeps sending the RSVP Path and Resv refresh messages to Device 4 and Device 5 so that they do not expire the state for the LSP; however, Device 3 suppresses these messages for Device 2.

Note A node restarts if it misses four ACKs or its hello src_instance (last source instance sent to its neighbor) changes so that its restart time = 0.

Before the restart time expires, Device 2 restarts and loads its configuration and graceful restart makes the configuration of Device 2 send the hello messages with a new source instance to all the data links attached. However, because Device 2 has lost the neighbor states, it does not know what destination instance it should use in those messages; therefore, all destination instances are set to 0.

When Device 3 sees the hello from Device 2, Device 3 stops the restart time for Device 2 and sends an ACK message back. When Device 3 sees a new source instance value in Device 2's hello message, Device 3 knows that Device 2 had a control plane failure. Device 2 gets Device 3's source instance value and uses it as the destination instance going forward.

Device 3 also checks the recovery time value in the hello message from Device 2. If the recovery time is 0, Device 3 knows that Device 2 was not able to preserve its forwarding information and Device 3 deletes all RSVP state that it had with Device 2.

If the recovery time is greater than 0, Device 1 sends Device 2 Path messages for each LSP that it had previously sent through Device 2. If these messages were previously refreshed in summary messages, they are sent individually during the recovery time. Each of these Path messages includes a Recovery_Label object containing the label value received from Device 2 before the failure.

When Device 3 receives a Path message from Device 2, Device 3 sends a Resv message upstream. However, Device 3 suppresses the Resv message until it receives a Path message.

How to Configure MPLS TE—RSVP Graceful Restart

This section describes how to configure MPLS TE—RSVP Graceful Restart.

Enabling Graceful Restart

To enable graceful restart, perform this procedure.

Note It is optional that you configure graceful restart on an interface.

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	**ip rsvp signalling hello graceful-restart mode help-neighbor** **Example:** `Device(config)# ip rsvp signalling hello graceful-restart mode help-neighbor`	Sets the number of DSCP hello messages on a neighboring device with restart capability.
Step 4	**interface** *type number* **Example:** `Device(config)# interface POS 1/0/0`	(Optional) Configures the interface type and number and enters interface configuration mode.
Step 5	**ip rsvp signalling hello graceful-restart** **Example:** `Device(config-if)# ip rsvp signalling hello graceful-restart`	(Optional) Enables RSVP TE graceful restart capability on a neighboring device.
Step 6	**exit** **Example:** `Device(config)# exit`	Exits to privileged EXEC mode.

Setting a DSCP Value

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 rsvp signalling hello graceful-restart dscp** *num* **Example:** `Device(config)# ip rsvp signalling hello graceful-restart dscp 30`	Sets the number of DSCP hello messages on a graceful restart-enabled Device.

	Command or Action	Purpose
Step 4	**end** **Example:** `Device(config)# end`	Exits to privileged EXEC mode.

Setting a Hello Refresh Interval

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 rsvp signalling hello graceful-restart refresh interval** *interval-value* **Example:** `Device(config)# ip rsvp signalling hello graceful-restart refresh interval 5000`	Sets a hello refresh interval on a device with graceful restart enabled.
Step 4	**end** **Example:** `Device(config)# end`	Exits to privileged EXEC mode.

Setting a Missed Refresh Limit

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	**ip rsvp signalling hello graceful-restart refresh misses** *msg-count* **Example:** `Device(config)# ip rsvp signalling hello graceful-restart refresh misses 5`	Sets a refresh limit on a device with graceful restart enabled.
Step 4	**end** **Example:** `Device(config)# end`	Exits to privileged EXEC mode.

Verifying Graceful Restart 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 rsvp hello graceful-restart** **Example:** `Device# show ip rsvp hello graceful-restart`	Displays information about the status of graceful restart and related parameters.
Step 3	**end** **Example:** `Device# end`	Exits to user EXEC mode.

Configuration Examples for MPLS TE—RSVP Graceful Restart

The following section provides configuration examples for MPLS TE—RSVP Graceful Restart.

Example: MPLS TE—RSVP Graceful Restart Example

In the following example, graceful restart is enabled, and related parameters, including a DSCP value, a refresh interval, and a missed refresh limit are set:

```
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# ip rsvp signalling hello graceful-restart mode help-neighbor
Device(config)# ip rsvp signalling hello graceful-restart dscp 30
Device(config)# ip rsvp signalling hello graceful-restart refresh interval 10000
Device(config)# ip rsvp signalling hello graceful-restart refresh misses 4
Device(config)# end
```

The following example verifies the status of graceful restart and the configured parameters:

```
Device# show ip rsvp hello graceful-restart
Graceful Restart:Enabled (help-neighbor only)
  Refresh interval:10000 msecs
  Refresh misses:4
  DSCP:0x30
  Advertised restart time:0 secs
  Advertised recovery time:0 secs
  Maximum wait for recovery:3600000 secs
```

Additional References

Related Documents

Related Topic	Document Title
RSVP commands: complete command syntax, command mode, defaults, usage guidelines, and examples	Cisco IOS Quality of Service Solutions Command Reference
Quality of service (QoS) classification	Classification Overview
QoS signalling	Signalling Overview
QoS congestion management	Congestion Management Overview
Stateful switchover	Stateful Switchover
MPLS Label Distribution Protocol	MPLS Label Distribution Protocol (LDP)
Information on stateful switchover, Cisco nonstop forwarding, graceful restart	NSF/SSO—MPLS TE and RSVP Graceful Restart
RSVP hello state timer	MPLS Traffic Engineering: RSVP Hello State Timer

Standards

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

MIBs

MIBs	MIBs Link
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 IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 3209	*RSVP-TE: Extensions to RSVP for LSP Tunnels*
RFC 3473	*Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions*
RFC 3478	*Graceful Restart Mechanism for Label Distribution*

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

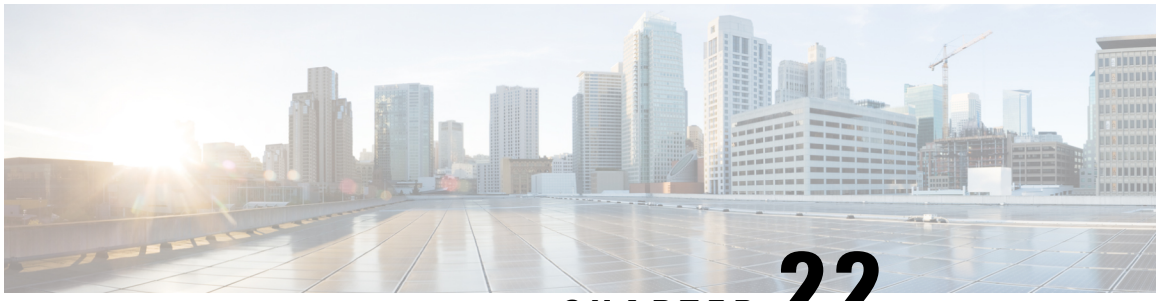
Feature History for MPLS Traffic Engineering—RSVP Graceful Restart

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 Cupertino 17.7.1	MPLS Traffic Engineering—RSVP Graceful Restart	The MPLS Traffic Engineering—RSVP Graceful Restart feature allows a neighboring Route Processor (RP) to recover from disruption in control plane service (specifically, the Label Distribution Protocol (LDP) component) without losing its Multiprotocol Label Switching (MPLS) forwarding state.

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/



CHAPTER 22

Configuring MPLS Traffic Engineering—Verbatim Path Support

Prerequisites for MPLS Traffic Engineering--Verbatim Path Support

- A Multiprotocol Label Switching (MPLS) TE tunnel must be configured globally.
- MPLS TE must be enabled on all links.

Restrictions for MPLS Traffic Engineering--Verbatim Path Support

- The **verbatim** keyword can be used only on a label-switched path (LSP) that is configured with the explicit path option.
- Reoptimization on the verbatim LSP is not supported.
- You cannot configure MPLS Traffic Engineering over the logical GRE tunnel interface.

Information About MPLS Traffic Engineering--Verbatim Path Support

MPLS TE LSPs usually require that all the nodes in the network are TE aware, meaning they have IGP extensions to TE in place. However, some network administrators want the ability to build TE LSPs to traverse nodes that do not support IGP extensions to TE, but that do support RSVP extensions to TE.

Verbatim LSPs are helpful when all or some of the intermediate nodes in a network do not support IGP extensions for TE.

When this feature is enabled, the IP explicit path is not checked against the TE topology database. Because the TE topology database is not verified, a Path message with IP explicit path information is routed using the shortest path first (SPF) algorithm for IP routing.

MPLS Traffic Engineering—Verbatim Path Support

The MPLS Traffic Engineering—Verbatim Path Support feature allows network nodes to support Resource Reservation Protocol (RSVP) extensions without supporting Interior Gateway Protocol (IGP) extensions for traffic engineering (TE), thereby bypassing the topology database verification process.

How to Configure MPLS Traffic Engineering—Verbatim Path Support

This section describes how to configure MPLS Traffic Engineering—Verbatim Path Support.

Configuring MPLS Traffic Engineering--Verbatim Path Support

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** *number* **Example:** `Device(config)# interface tunnel 1`	Configures a tunnel interface and enters interface configuration mode. • The *number* argument identifies the tunnel number to be configured.

	Command or Action	Purpose
Step 4	**ip unnumbered loopback** *number* **Example:** `Device(config-if)# ip unnumbered loopback 1`	Configures an unnumbered IP interface, which enables IP processing without an explicit address. A loopback interface is usually configured with the Device ID. **Note** An MPLS traffic engineering tunnel interface should be unnumbered because it represents a unidirectional link.
Step 5	**tunnel destination** {*host-name*| *ip-address*} **Example:** `Device(config-if)# tunnel destination 10.100.100.100`	Specifies the destination for a tunnel. • The *host-name* argument is the name of the host destination. • The *ip-address* argument is the IP Version 4 address of the host destination expressed in decimal in four-part, dotted notation.
Step 6	**tunnel mode mpls traffic-eng** **Example:** `Device(config-if)# tunnel mode mpls traffic-eng`	Sets the tunnel encapsulation mode to MPLS traffic engineering.
Step 7	**tunnel mpls traffic-eng bandwidth** {**sub-pool** *kbps* | *kbps}* **Example:** `Device(config-if)# tunnel mpls traffic-eng bandwidth 1000`	Configures the bandwidth required for an MPLS TE tunnel and assigns it either to the sub-pool or the global pool. • The **sub-pool** keyword indicates a subpool tunnel. • The *kbps* argument is the bandwidth, in kilobits per second, set aside for the MPLS TE tunnel. The range is from 1 to 4294967295.
Step 8	**tunnel mpls traffic-eng autoroute announce** **Example:** `Device(config-if)# tunnel mpls traffic-eng autoroute announce`	Specifies that IGP should use the tunnel (if the tunnel is up) in its enhanced SPF calculation.
Step 9	**tunnel mpls traffic-eng priority** *setup-priority* [*hold-priority*] **Example:** `Device(config-if)# tunnel mpls traffic-eng priority 1 1`	Configures setup and reservation priority for a tunnel. • The *setup-priority* argument is the priority used when signaling an LSP for this tunnel to determine which existing tunnels can be preempted.

	Command or Action	Purpose
		Valid values are from 0 to 7. A lower number indicates a higher priority. An LSP with a setup priority of 0 can preempt any LSP with a non-0 priority. • The *hold-priority* argument is the priority associated with an LSP for this tunnel to determine if it should be preempted by other LSPs that are being signaled. Valid values are from 0 to 7, where a lower number indicates a higher priority.
Step 10	**tunnel mpls traffic-eng path-option** *preference-number* {**dynamic** [**attributes** *string* | **bandwidth** {**sub-pool** *kbps* | *kbps*} | **lockdown | verbatim**] | **explicit**{**name** *path-name* | **identifier** *path-number* **}}** **Example:** `Device(config-if)# tunnel mpls traffic-eng path-option 1 explicit name test verbatim` **Example:**	Specifies LSP-related parameters, including the **verbatim** keyword used with an explicit path option, for an MPLS TE tunnel. • The *preference-number* argument identifies the path option. • The **protect** keyword and *preference-number argument identify the path option with protection.* • The **dynamic** keyword indicates that the path option is dynamically calculated. (The Device figures out the best path.) • The **explicit** keyword indicates that the path option is specified. The IP addresses are specified for the path. • The **name** *path-name*keyword argument combination identifies the name of the explicit path option. • The *path-number* argument identifies the number of the explicit path option. • The **verbatim** keyword bypasses the topology database verification. **Note** You can use the **verbatim** keyword only with the explicit path option. • The **attributes** *string* keyword argument combination names an attribute list to specify path options for the LSP. • The **bandwidth** keyword specifies the LSP bandwidth.

	Command or Action	Purpose
		• The **sub-pool** keyword indicates a subpool path option. • The *kbps* argument is the number of kilobits per second set aside for the path option. The range is from 1 to 4294967295. • The **lockdown** keyword disables reoptimization of the LSP.
Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 12	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Verifying Verbatim LSPs for MPLS TE Tunnels

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**show mpls traffic-eng tunnels** *tunnel-interface* [**brief**] **Example:** `Device# show mpls traffic-eng tunnels tunnel1`	Displays information about tunnels including those configured with an explicit path option using verbatim.
Step 3	**disable** **Example:** `Device# disable`	(Optional) Exits to user EXEC mode.

Configuration Examples for MPLS Traffic Engineering—Verbatim Path Support

The following section provides configuration examples for MPLS Traffic Engineering—Verbatim Path Support.

Example: Configuring MPLS Traffic Engineering: Verbatim Path Support

The following example shows a tunnel that has been configured with an explicit path option using verbatim:

```
interface tunnel 1
 ip unnumbered loopback 1
 tunnel destination 10.10.100.100
 tunnel mode mpls traffic-eng
 tunnel mpls traffic-eng bandwidth 1000
 tunnel mpls traffic-eng autoroute announce
 tunnel mpls traffic-eng priority 1 1
 tunnel mpls traffic-eng path-option 1 explicit name path1 verbatim
```

Additional References

Related Documents

Related Topic	Document Title
MPLS Label Distribution Protocol	MPLS Label Distribution Protocol (LDP) feature module

Standards

Standard	Title
No new or modified standards are supported by this feature, and 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 IOS 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 release.	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

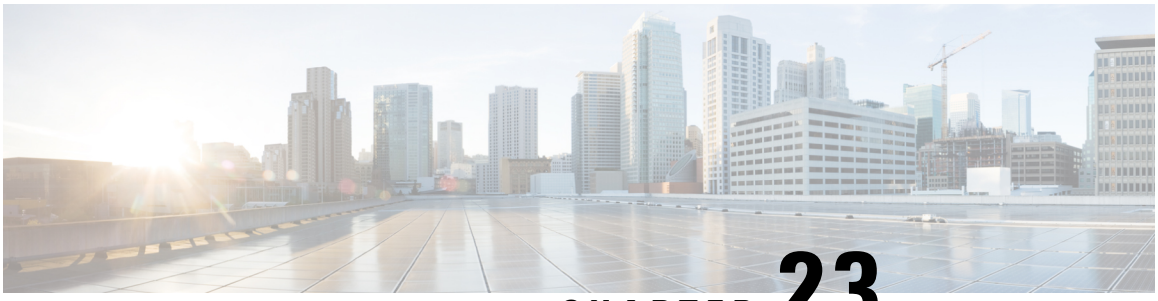
Feature History for MPLS Traffic Engineering Verbatim Path Support

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 Cupertino 17.7.1	MPLS Traffic Engineering—Verbatim Path Support	The MPLS Traffic Engineering—Verbatim Path Support feature allows network nodes to support Resource Reservation Protocol (RSVP) extensions without supporting Interior Gateway Protocol (IGP) extensions for traffic engineering (TE), thereby bypassing the topology database verification process.

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/



CHAPTER 23

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

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.
- On a Cisco StackWise Virtual Multichassis EtherChannel configured on a VPLS network that supports IGMP snooping, if the number of IGMP join requests exceed 12000, and also if a changeover happens, then a traffic drop occurs for around 40 seconds after the standby switch joins back on the Cisco StackWise Virtual.

Information About VPLS, VPLS BGP-Based Autodiscovery and Flow-Aware Transport

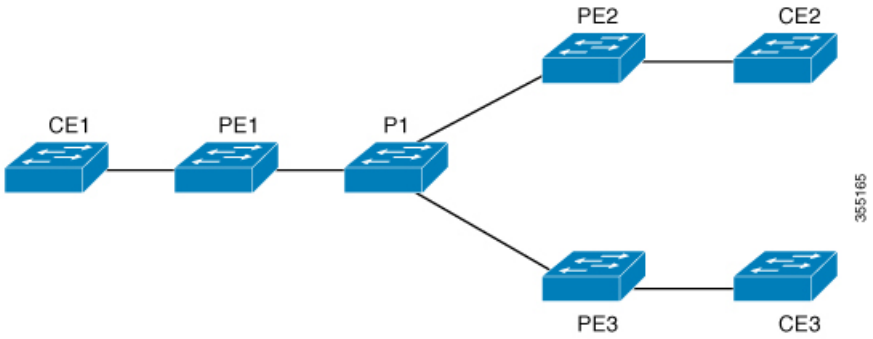
The following sections provide information about VPLS, VPLS BGP-based autodiscovery and flow-aware transport.

VPLS Overview

VPLS enables enterprises to link together their Ethernet-based LANs from multiple sites through the infrastructure provided by their service provider. From the enterprise perspective, the service provider's public network looks like one large 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.

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

Figure 26: VPLS Topology



About Full-Mesh Configuration

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

For a full-mesh configuration, a virtual forwarding instance (VFI) is required on each participating PE device. The VFI includes 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 device.

A VPLS instance constitutes a set of VFIs formed by the interconnection of the emulated VCs. The VPLS instance forms the logic bridge over the 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 the static configuration using the Cisco IOS CLI.

The full-mesh configuration allows the PE device to maintain a single broadcast domain. So when the PE device 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 the 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. The split-horizon principle ensures that a packet received on an emulated VC 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 VFI of a particular VPLS domain.

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

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

About VPLS BGP-Based Autodiscovery

VPLS autodiscovery enables each 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. With VPLS autodiscovery enabled, it is no longer needed 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 (PWs) in a VPLS domain.

BGP uses the Layer 2 VPN Routing Information Base (RIB) to store endpoint provisioning information, which is updated each time any Layer 2 VFI is configured. The prefix and path information is stored in the Layer 2 VPN 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 Layer 2 VPN-based services.

The BGP autodiscovery mechanism facilitates the configuration of Layer 2 VPN 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 MPLS network.

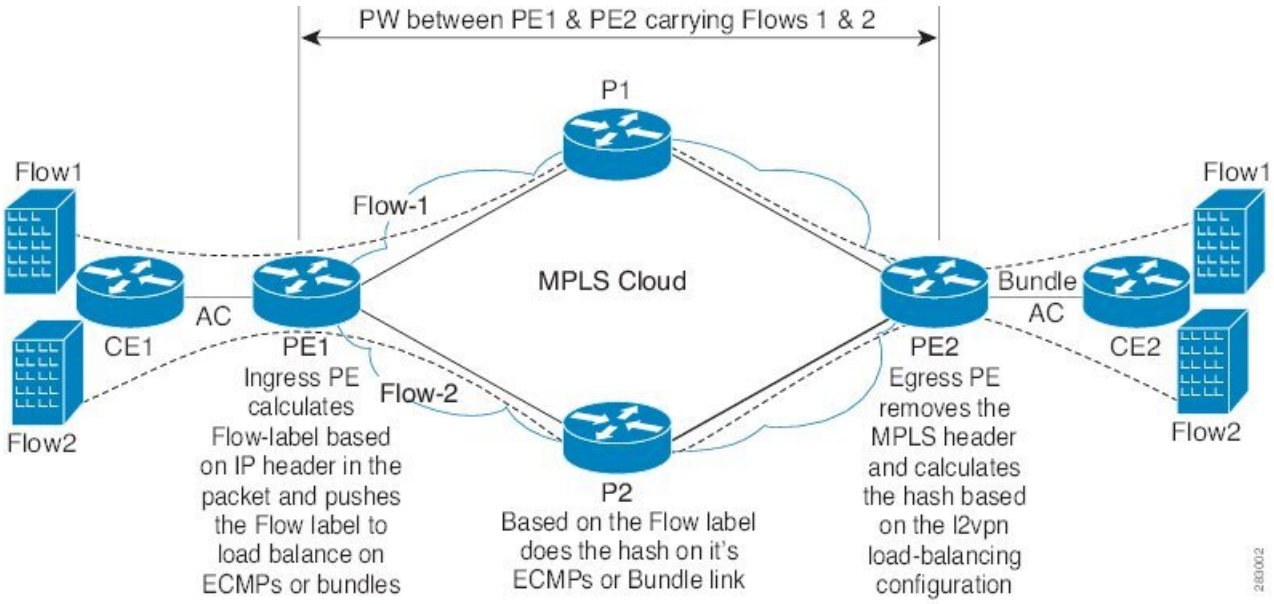
About Flow-Aware Transport Pseudowire

Devices typically load-balance traffic based on the lower most label in the label stack which is the same label for all flows on a given pseudowire. This can lead to asymmetric loadbalancing. The flow, in this context, refers to a sequence of packets that have the same source and destination pair. The packets are transported from a source provider edge (PE) device to a destination PE device.

Flow-aware transport PWs provide the capability to identify individual flows within a PW and provide devices the ability to use these flows to load-balance traffic. Flow-aware transport PWs are used to load-balance traffic in the core when equal cost multipaths (ECMP) are used. A flow label is created based on individual packet flows entering a PW; and is inserted as the lower most label in the packet. Devices can use the flow label for load-balancing which provides a better traffic distribution across ECMP paths or link-bundled paths in the core.

Figure 27: Flow-aware transport PW with two flows distributing over ECMPs and Bundle-Links shows a flow-aware transport PW with two flows distributing over ECMPs and bundle links.

Figure 27: Flow-aware transport PW with two flows distributing over ECMPs and Bundle-Links



An extra label is added to the stack, called the flow label, which contains the flow information of a virtual circuit (VC). A flow label is a unique identifier that distinguishes a flow within the PW, and is derived from source and destination MAC addresses, and source and destination IP addresses. The flow label contains the end of label stack (EOS) bit set and inserted after the VC label and before the control word (if any). The ingress PE calculates and forwards the flow label. The flow-aware transport PW configuration enables the flow label. The egress PE discards the flow label such that no decisions are made.

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

Flow-aware transport PW works based on port-channel load-balance algorithm only.

Interoperability Between Cisco Catalyst 6000 Series Switches and Cisco Catalyst 9000 Series Switches

The following section describes how to enable sending and receiving flow labels between Cisco Catalyst 6000 Series Switches and Cisco Catalyst 9000 Series Switches.

On a Cisco Catalyst 6000 Series Switch configured with flow-aware transport PW (using Advanced VPLS) flow label negotiations are not supported. If the Cisco Catalyst 6000 Series Switch is in interoperability with a remote PE device such as a Cisco Catalyst 9000 Series Switch, then the Cisco Catalyst 9000 Series Switch cannot receive and send the flow label for data traffic. Configuring the **load-balance flow-label both static** command on the Cisco Catalyst 9000 Series Switch allows the Cisco Catalyst 9000 Series Switch to receive and send the flow labels even though the Cisco Catalyst 6000 Series Switch does not support flow label negotiations.

The following is a configuration example to enable sending and receiving flow labels:

```
Device> enable
Device# configure terminal
```

```
Device(config)# template type pseudowire mpls
Device(config-template)# encapsulation mpls
Device(config-template# load-balance flow ip dst-ip
Device(config-template)# load-balance flow-label both static
Device(config-template# end
```

IGMP/MLD Snooping over VPLS

Starting with the Cisco IOS XE Bengaluru 17.6.1 release, support for MLD Snooping over VPLS has been introduced. Support for IGMP Snooping over VPLS was introduced in Cisco IOS XE Amsterdam 17.1.1 release.

When you enable IGMP/MLD Snooping over VPLS, traffic is forwarded on pseudowires that receive IGMP/MLD reports from remote Provider Edge (PE) devices. IGMP/MLD queries and reports are flooded to all the pseudowires.

MLD Snooping is not enabled by default and needs to be configured at the global level.

IGMP snooping is enabled by default at the global level.

For more information on MLD Snooping, see Information About Configuring IPv6 MLD Snooping.

For more information on IGMP Snooping, see IGMP Snooping.

How to Configure VPLS, VPLS BGP-Based Autodiscovery and Flow-Aware Transport

The following sections provide configuration information about VPLS, VPLS BGP-based autodiscovery and flow-aware transport.

Configuring Layer 2 PE Device Interfaces to CE Devices

You must configure Layer 2 PE device interfaces to CE devices. The following sections provide various configuration tasks that need to be completed before configuring VPLS.

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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
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.
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:** `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.
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.
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 VPLS

VPLS can be configured using either the Xconnect mode or protocol-CLI method.The following sections provide information about how to configure VPLS.

Configuring VPLS in Xconnect Mode

The following sections provide information on configuring VPLS in Xconnect 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:** `Device(config)# mpls label protocol ldp`	Specifies the default Label Distribution Protocol (LDP) for a platform.
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.

	Command or Action	Purpose
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*.
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.

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

Configuring VPLS in Protocol-CLI Mode

The following sections provide information on configuring VPLS in protocol-CLI mode.

Configuring VPLS in Protocol-CLI Mode

To configure VPLS in protocol-CLI mode, 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	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.

	Command or Action	Purpose
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**member** *ip-address* **encapsulation mpls** **Example:** `Device(config-vfi)# member 2.2.2.2 encapsulation mpls`	Specifies the devices that form a point-to-point Layer 2 VPN VFI connection.
Step 6	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 7	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	OR
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 8	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 9	**end** **Example:** `Device(config-vlan-config)# end`	Returns to privileged EXEC mode.

Configuring VPLS Flow-Aware Transport with Pseudowire Interface (in Protocol-CLI Mode)

To configure VPLS flow-aware transport with pseudowire interface, 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	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 1001`	Establishes a PW with the specified name, and enters pseudowire interface configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-if)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)# neighbor 10.1.1.200 200`	Specifies the peer IP address and VC ID value of a Layer 2 VPN PW.
Step 6	**load-balance flow** **Example:** `Device(config-if)# load-balance flow`	Enables the load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 7	**load-balance flow-label** **Example:** `Device(config-if)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 8	**exit** **Example:** `Device(config-if)# exit`	Exits to privileged EXEC mode.
Step 9	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 10	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.

	Command or Action	Purpose
Step 11	**member pseudowire** *number* **Example:** `Device(config-vfi)# member pseudowire 1001`	Adds the pseudowire interface as a member of the VFI.
Step 12	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 13	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 14	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 15	**end** **Example:** `Device(config-vlan-config)# end`	Returns to privileged EXEC mode.

Configuring VPLS Flow-Aware Transport Using a Template (in Protocol-CLI Mode)

Configuring VPLS flow-aware transport using a template allows multiple PWs to share the same configuration.

To configure VPLS flow-aware transport using a template, 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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**template type pseudowire** [*template-name*] **Example:** `Device(config)# template type pseudowire mpls`	Specifies the name of a Layer 2 PW and enters pseudowire template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**load-balance flow** **Example:** `Device(config-template)# load-balance flow`	Enables the load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 6	**load-balance flow-label** **Example:** `Device(config-template)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits to privileged EXEC mode.
Step 8	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 9	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 10	**member** *ip-address* **template** *template-name* **Example:** `Device(config-vfi)# member 102.102.102.102 template mpls`	Specifies the devices that form a point-to-point Layer 2 VPN VFI connection. • **ip-address**: IP address of the VFI neighbor.

	Command or Action	Purpose
		• **template** *template-name*: Specifies the template name mpls as the template method.
Step 11	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 12	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 13	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 14	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

Configuring VPLS Flow-Aware Transport Using Pseudowire and a Template (in Protocol-CLI Mode)

To configure VPLS flow-aware transport using both PW and a template, 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.

	Command or Action	Purpose
Step 3	**template type pseudowire** [*template-name*] **Example:** `Device(config)# template type pseudowire mpls`	Specifies the name of a Layer 2 PW and enters pseudowire template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**load-balance flow** **Example:** `Device(config-template)# load-balance flow`	Enables the load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 6	**load-balance flow-label** **Example:** `Device(config-template)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits to privileged EXEC mode.
Step 8	**interface pseudowire** *number* **Example:** `Device(config)# interface pseudowire 1001`	Establishes a PW with the specified name, and enters pseudowire interface configuration mode.
Step 9	**source template type pseudowire** [*template-name*] **Example:** `Device(config-if)# source template type pseudowire mpls`	Configures the source template of type pseudowire named mpls.
Step 10	**neighbor** *peer-address vcid-value* **Example:** `Device(config-if)# neighbor 10.1.1.200 200`	Specifies the peer IP address and VC ID value of a Layer 2 VPN PW.

	Command or Action	Purpose
Step 11	**exit** **Example:** `Device(config-if)# exit`	Exits to privileged EXEC mode.
Step 12	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 13	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 14	**member pseudowire** *number* **Example:** `Device(config-vfi)# member pseudowire 1001`	Adds the pseudowire interface as a member of the VFI.
Step 15	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 16	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	OR
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 17	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 18	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

Configuring VPLS BGP-based Autodiscovery

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

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:** `Device(config-vfi)# vpn id 2128`	Configures a VPN ID for the VPLS domain.
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:** `Device(config-router)# no bgp default ipv4-unicast`	Disables the IPv4 unicast address family for the BGP routing process. **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 or Action	Purpose
		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:** `Device(config-router)# address-family l2vpn vpls`	Specifies the Layer 2 VPN address family and enters address family configuration mode. 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:**	Exits address family configuration mode and returns to router configuration mode.

	Command or Action	Purpose
	`Device(config-router-af)# exit-address-family`	
Step 14	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.

Configuring VPLS BGP-based Autodiscovery in Protocol-CLI Mode

The following sections provide information on configuring VPLS BGP-based autodiscovery in protocol-CLI mode.

Configuring VPLS BGP based Autodiscovery in Protocol-CLI mode

To configure VPLS BGP based autodiscovery in protocol-CLI mode, 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	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 4	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 5	**autodiscovery bgp signaling ldp** **Example:** `Device(config-vfi)# autodiscovery bgp signaling ldp`	Enables BGP signaling and LDP signaling.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Device(config-vfi-autodiscovery)# exit`	Exits to privileged EXEC mode.
Step 7	**exit** **Example:** `Device(config-vfi)# exit`	Exits to privileged EXEC mode.

Step 8	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 9	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 10	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

Configuring VPLS BGP based Autodiscovery Flow-Aware Transport using Template (in Protocol-CLI Mode)

To configure VPLS BGP based autodiscovery flow-aware transport using template, 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.

	Command or Action	Purpose
Step 3	**template type pseudowire** [*template-name*] **Example:** `Device(config)# template type pseudowire mpls`	Specifies the name of a Layer 2 PW and enters pseudowire template configuration mode.
Step 4	**encapsulation mpls** **Example:** `Device(config-template)# encapsulation mpls`	Specifies the tunneling encapsulation as MPLS.
Step 5	**load-balance flow** **Example:** `Device(config-template)# load-balance flow`	Enables the Any Transport over MPLS (AToM) load balancing with PW feature so that load balancing is done on a per-flow basis.
Step 6	**load-balance flow-label** **Example:** `Device(config-template)# load-balance flow-label both`	Enables the flow-aware transport of MPLS PW feature and specifies how flow labels are to be used.
Step 7	**exit** **Example:** `Device(config-template)# exit`	Exits to privileged EXEC mode.
Step 8	**l2vpn vfi context** *vfi-name* **Example:** `Device(config)# l2vpn vfi context vpls1`	Establishes an Layer 2 VPN VFI context and enters Layer 2 VFI configuration mode.
Step 9	**vpn id** *vpn-id* **Example:** `Device(config-vfi)# vpn id 10`	Configures a VPN ID for the VPLS domain.
Step 10	**autodiscovery bgp signaling ldp template** *name* **Example:** `Device(config-vfi)# autodiscovery bgp signaling ldp template mpls`	Enables BGP signaling and LDP signaling.
Step 11	**exit** **Example:**	Exits to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-vfi)# exit`	

Step 12	Select one of the following:
	• **vlan configuration** *vlan-id*
	• **interface vlan** *vlan-id*
	Example:
	`Device(config)# vlan configuration 100`
	`OR`
	`Device(config)# interface vlan 100`
		Applies configuration to be applied on the VLAN or interface and enters VLAN or interface configuration mode.

Step 13	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpls1`	Binds a VFI instance to a VLAN or an interface.
Step 14	**end** **Example:** `Device(config-vlan-config)# end`	Exits to privileged EXEC mode.

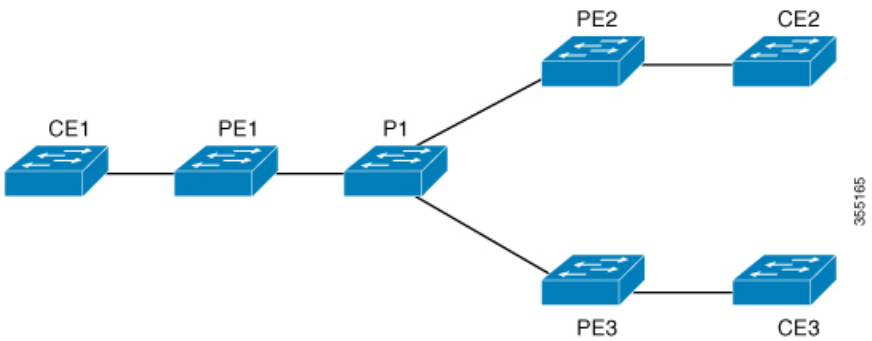
Configuration Examples for VPLS and VPLS BGP-Based Autodiscovery

This section provides the configuration examples for VPLS and VPLS BGP-Based Autodiscovery.

Example: Configuring VPLS in Xconnect Mode

The following example shows how to configure VPLS on a PE1 and PE2 devices:

Figure 28: VPLS Topology



PE1 Configuration

```
Device> enable
Device# configure terminal
Device(config)# pseudowire-class vpls2129
Device(config-if)# encapsulation mpls
Device(config-if)# exit
Device(config)# l2 vfi 2129 manual
Device(config-vfi)# vpn id 2129
Device(config-vfi)# neighbor 44.254.44.44 pw-class vpls2129
Device(config-vfi)# neighbor 188.98.89.98 pw-class vpls2129
Device(config-vfi)# exit
Device(config)# interface TenGigabitEthernet1/0/24
Device(config-if)# switchport trunk allowed vlan 2129
Device(config-if)# switchport mode trunk
Device(config-if)# exit
Device(config)# interface vlan 2129
Device(config-vlan-config)# no ip address
Device(config-vlan-config)# xconnect vfi 2129
```

Examples: Verifying VPLS Configured in Xconnect Mode

The following example is a sample output of the **show mpls l2transport vc detail** command. This command provides information about the virtual circuits.

```
Device# show mpls l2transport vc detail
 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 following example is a sample output of the **show l2vpn atom vc** command. The command shows that AToM over MPLS is configured on a VC.

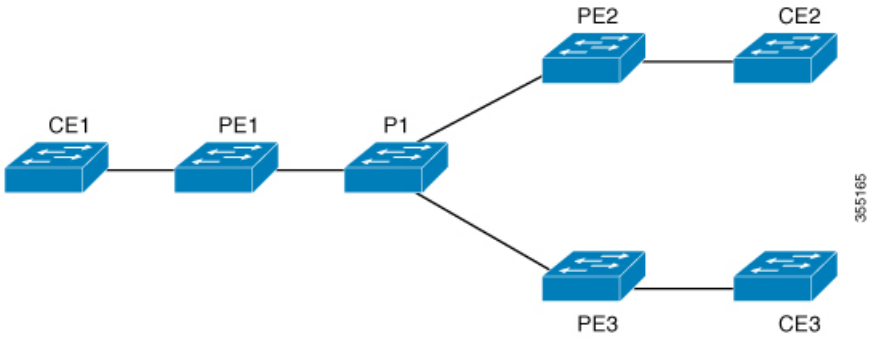
```
Device# show l2vpn atom vc detail

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
```

Example: Configuring VPLS Flow-Aware Transport Using a Template (in Protocol-CLI Mode)

The following example shows how to configure VPLS on a PE1 and PE2 devices:

Figure 29: VPLS Topology



PE1 Configuration

```
Device> enable
Device# configure terminal
Device(config)# template type pseudowire mpls
Device(config-template)# encapsulation mpls
Device(config-template)# load-balance flow ip dst-ip
Device(config-template)# load-balance flow-label both
Device(config-template)# exit
Device(config)# interface Loopback0
Device(config-if)# ip address 1.1.1.30 255.255.255.255
Device(config-if)# ip ospf 1 area 0
Device(config-if)# exit
Device(config)# interface TwentyFiveGigE1/0/9
Device(config-if)# no switchport
Device(config-if)# ip address 80.0.0.30 255.255.255.0
Device(config-if)# ip ospf 1 area 0
Device(config-if)# mpls ip
Device(config-if)# exit
Device(config)# l2vpn vfi context foo
Device(config-vfi)# vpn id 2129
Device(config-vfi)# member 1.1.1.20 template mpls
Device(config-vfi)# exit
Device(config)# interface TwentyFiveGigE1/0/2
Device(config-if)# switchport mode access
Device(config-if)# switchport access vlan 100
Device(config-if)# exit
Device(config)# interface vlan 100
Device(config-vlan-config)# member vfi foo
Device(config-vlan-config)# end
```

Example: Configuring VPLS BGP-Auto Discovery

The following example shows how to configure VPLS on a PE device:

```
Device> enable
Device# configure terminal
Device(config)# router bgp 1000
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# bgp graceful-restart
Device(config-router)# neighbor 44.254.44.44 remote-as 1000
Device(config-router)# neighbor 44.254.44.44 update-source Loopback300
Device(config-router)# address-family l2vpn vpls
Device(config-router-af)# neighbor 44.254.44.44 activate
Device(config-router-af)# neighbor 44.254.44.44 send-community both
Device(config-router-af)# exit-address-family
Device(config-router-af)# end
Device(config)# l2 vfi 2128 autodiscovery
Device(config-vfi)# vpn id 2128
Device(config-vfi)# exit
Device(config)# interface vlan 2128
Device(config-vlan-config)# no ip address
Device(config-vlan-config)# xconnect vfi 2128
!
```

Example: Verifying VPLS BGP-Auto Discovery

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

```
Device# show platform software fed sw 1 matm macTable vlan 2000

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 example is a sample output of the **show bgp l2vpn vpls all** command.

```
Device# show bgp l2vpn vpls all

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 History for VPLS and VPLS BGP-Based Autodiscovery

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

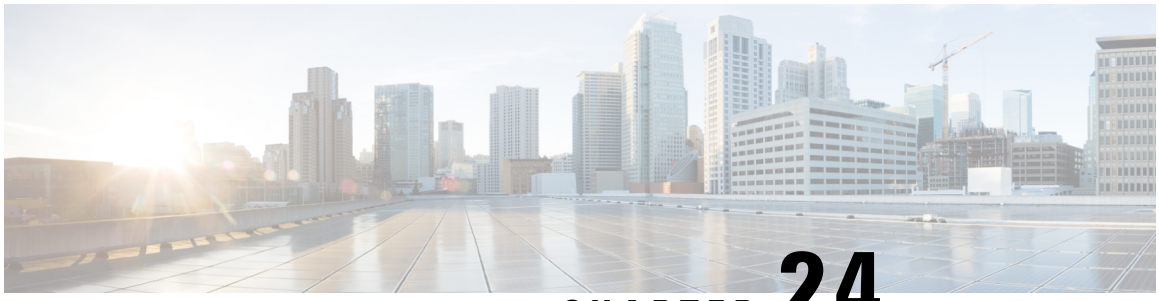
These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.6.1	Configuring VPLS	VPLS enables enterprises to link together their Ethernet-based LANs from multiple sites via the infrastructure provided by their service provider.
Cisco IOS XE Gibraltar 16.12.1	Configuring VPLS BGP-based Autodiscovery	VPLS Autodiscovery enables each PE device to discover other PE devices that are part of the same VPLS domain.
Cisco IOS XE Amsterdam 17.1.1	VPLS Layer 2 Snooping : IGMP (IPv4)	IGMP snooping is supported on a VPLS configured network.
Cisco IOS XE Bengaluru 17.6.1	MLD Snooping over VPLS	Support was introduced for MLD Snooping over VPLS. It allows traffic to be forwarded on pseudowires that receive IGMP/MLD reports from remote Provider Edge (PE) devices.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.7.1	VPLS BGP-based Autodiscovery	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 24

Configuring Hierarchical VPLS with MPLS Access

Configuring Virtual Private LAN Service (VPLS) requires a full mesh of tunnel label switched paths (LSPs) between all the provider edge (PE) devices that participate in the VPLS. With full-mesh configuration, signaling overhead and packet replication requirements for each provisioned virtual circuit (VC) on a PE device are high. Configuring Hierarchical VPLS with Multiprotocol Label Switching (MPLS) Access reduces signaling overhead and packet replication between devices.

Prerequisites for Configuring Hierarchical VPLS with MPLS Access

Configure the PE to customer edge (CE) interface with a list of allowed VLANs.

Restrictions for Configuring Hierarchical VPLS with MPLS Access

- This feature is not supported if VPLS Autodiscovery is configured on pseudowires (PWs) that are attached to user provider edge (U-PE) devices. (When you create the VPLS, you can manually create the virtual forwarding interface (VFI)).
- This feature is not supported if Q-in-Q access is configured between a U-PE device and a N-PE device.
- Internet Group Management Protocol (IGMP) snooping is not supported.
- Cisco Discovery Protocol (CDP) is not supported.

- Multiprotocol Label Switching (MPLS) over generic routing encapsulation (GRE) and VPLS over GRE are not supported.

Information About Configuring Hierarchical VPLS with MPLS Access

The following section provides information about configuring hierarchical VPLS with MPLS access.

About Hierarchical VPLS with MPLS Access

A standard VPLS configuration comprises CE devices and PE devices. Using the Hierarchical VPLS with MPLS Access feature, each PE device is replaced with a U-PE and an N-PE device. U-PE devices communicate with the CE devices and N-PE devices on the access side, and N-PE devices communicate with other N-PE devices on the provider core.

Figure 30: Hierarchical VPLS with MPLS Access Configuration shows a hierarchical VPLS with MPLS access configuration. Each CE device is connected to a U-PE device through an attachment circuit. A U-PE device is connected to an N-PE device through a single pseudowire (PW) for each VPLS instance.

The following configuration types are supported between a U-PE device and an N-PE device:

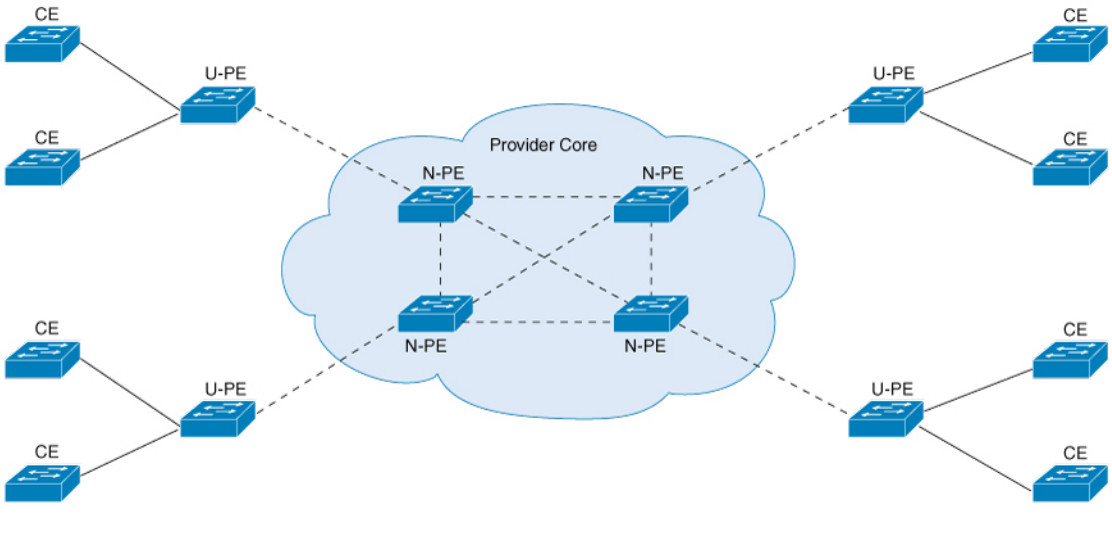
- Ethernet Q-in-Q

Note Ethernet Q-in-Q configurations are not supported in Cisco IOS XE Amsterdam 17.2.x.

- EoMPLS

N-PE devices are connected to each other through a mesh of PWs. Packets from a U-PE device to an N-PE device can be forwarded to other U- PE devices that are connected to the same N-PE device and to other N-PE devices, if any, because split horizon is disabled. Packets in the provider core are not forwarded back to the provider core because split horizon is enabled.

Figure 30: Hierarchical VPLS with MPLS Access Configuration



Features that Support Hierarchical VPLS with MPLS Access Configuration

The following is a list of features that support the Hierarchical VPLS with MPLS Access Configuration:

- VPLS integrated routing and bridging (IRB)
- VPLS MAC address withdrawal
- PW redundancy
- VPLS flow-aware transport PW

How to Configure Hierarchical VPLS with MPLS Access

The following sections provide information on how to configure the Hierarchical VPLS with MPLS Access feature.

Configuring VPLS (Protocol-CLI Method) on an N-PE Device

To configure VPLS (Protocol-CLI method) on an N-PE device, perform this procedure,

Note Repeat this procedure on each N-PE device.

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	**l2vpn vfi context** *name* **Example:** `Device(config)# l2vpn vfi context vpn100`	Establishes a Layer 2 VPN VFI between two or more separate networks, and enters L2VFI configuration mode.
Step 4	**vpn id** *vpn id* **Example:** `Device(config-vfi)# vpn id 100`	Sets a VPN ID on the VPLS instance. • Use the same VPN ID for the N-PE devices that belong to the same VPN. • Make sure that the VPN ID is unique for each VPN in the service provider network. The range is from 1 to 4294967295.
Step 5	**member** *ip-address* **encapsulation mpls** **Example:** `Device(config-vfi)# member 4.4.4.4 encapsulation mpls`	Specifies the device that forms a point-to-point L2VPN VFI connection. • *ip-address*: IP address of the VFI neighbor (the N-PE device). • **encapsulation mpls**: Specifies **mpls** as the data encapsulation method.
Step 6	**exit** **Example:** `Device(config-vlan-config)# exit`	Returns to global configuration mode.
Step 7	**vlan configuration** *vlan-id* **Example:** `Device(config)# vlan configuration 100`	Applies the configuration on the VLAN, and enters VLAN configuration mode.
Step 8	**member vfi** *vfi-name* **Example:** `Device(config-vlan-config)# member vfi vpn100`	Binds a VFI instance to a VLAN or an interface.

	Command or Action	Purpose
Step 9	**member** *ip-address* **encapsulation mpls** **Example:** `Device(config-vlan-config)# member 19.19.19.19 encapsulation mpls`	Specifies the device that forms a point-to-point L2VPN VFI connection. • *ip-address*: IP address of the VFI neighbor (the U-PE device). • **encapsulation mpls**: Specifies **mpls** as the data encapsulation method.
Step 10	**end** **Example:** `Device(config-vlan-config)# end`	Exits privileged EXEC mode.

Configuring EoMPLS VLAN (Xconnect Method) on an U-PE Device

To configure EoMPLS VLAN (Xconnect method) on an U-PE device, perform this procedure,

Note Perform this task on each U-PE device

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.subinterface* **Example:** `Device(config)# interface TenGigabitEthernet1/6/21.100`	Defines the subinterface to be configured, and enters subinterface configuration mode.
Step 4	**encapsulation dot1q** *vlan-id* **Example:** `Device(config-subif)# encapsulation dot1q 100`	Enables IEEE 802.1Q encapsulation of traffic on the subinterface.

	Command or Action	Purpose
Step 5	**xconnect** *peer-ip-addr vc-id* **encapsulation mpls** **Example:** `Device(config-if)# xconnect 3.3.3.3 150 encapsulation mpls`	Binds the attachment circuit to a PW VC. The syntax for this command is the same as for all the other Layer 2 transports.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.

Configuration Examples for Hierarchical VPLS with MPLS Access

The following example shows how to configure loopback interface for N-PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface loopback 0
Device(config-if)# ip address 3.3.3.3 255.255.255.255
Device(config-if)# ip ospf 1 area 0
Device(config-if)# end
```

The following example shows how to enable MPLS on N-PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface For 1/0/20
Device(config-if)# ip address 17.0.0.2 255.255.255.0
Device(config-if)# mpls ip
Device(config-if)# ip ospf 1 area 0
Device(config-if)# end
```

The following example shows how to enable VFI on N-PE1:

```
Device> enable
Device# configure terminal
Device(config)# l2vpn vfi context vpn100
Device(config-vfi)# vpn id 100
Device(config-vfi)# member 4.4.4.4 encapsulation mpls
```

The following example shows how to specify a point-to-point Layer 2 VPN (L2VPN) VFI connection on N-PE1:

```
Device> enable
Device# configure terminal
Device(config)# vlan configuration 100
Device(config-vlan-config)# member vfi vpn100
Device(config-vlan-config)# mmember 19.19.19.19 encapsulation mpls
```

The following example shows how to configure loopback interface for N-PE2:

```
Device> enable
Device# configure terminal
Device(config)# interface loopback 0
```

```
Device(config-if)# ip address 4.4.4.4 255.255.255.255
Device(config-if)# ip ospf 1 area 0
Device(config-if)# end
```

The following example shows how to enable MPLS on N-PE2:

```
Device> enable
Device# configure terminal
Device(config)# interface For 1/0/5
Device(config-if)# ip address 13.0.0.2 255.255.255.0
Device(config-if)# mpls ip
Device(config-if)# ip ospf 1 area 0
Device(config-if)# end
```

The following example shows how to enable VFI on the N-PE2:

```
Device> enable
Device# configure terminal
Device(config)# l2vpn vfi context vpn100
Device(config-vfi)# vpn id 100
Device(config-vfi)# member 3.3.3.3 encapsulation mpls
```

The following example shows how to specify a point-to-point L2VPN VFI connection on N-PE2:

```
Device> enable
Device# configure terminal
Device(config)# vlan configuration 100
Device(config-vlan-config)# member vfi vpn100
```

The following example shows how to configure loopback interface for U-PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface loopback 0
Device(config-if)# ip address 19.19.19.19 255.255.255.255
Device(config-if)# ip ospf 1 area 0
Device(config-if)# end
```

The following example shows how to enable MPLS on U-PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface Forty2/1
Device(config-if)# ip address 17.0.0.1 255.255.255.0
Device(config-if)# mpls ip
Device(config-if)# ip ospf 1 area 0
Device(config-if)# end
```

The following example shows how to enable EoMPLS on U-PE1:

```
Device> enable
Device# configure terminal
Device(config)# interface TenGig6/21.100
Device(config-if)# encapsulation dot1q 100
Device(config-if)# xconnect 3.3.3.3 100 encapsulation mpls
```

Additional References for Configuring Hierarchical VPLS with MPLS Access

Related Documents

Related Topic	Document Title
Configuring EoMPLS in VLAN mode (Protocol-CLI method)	Configuring Ethernet-over-MPLS (EoMPLS)
Configuring VPLS and VPLS flow-aware transport	Configuring Virtual Private LAN Service (VPLS) and VPLS BGP-Based Autodiscovery

Feature History for Configuring Hierarchical VPLS with MPLS Access

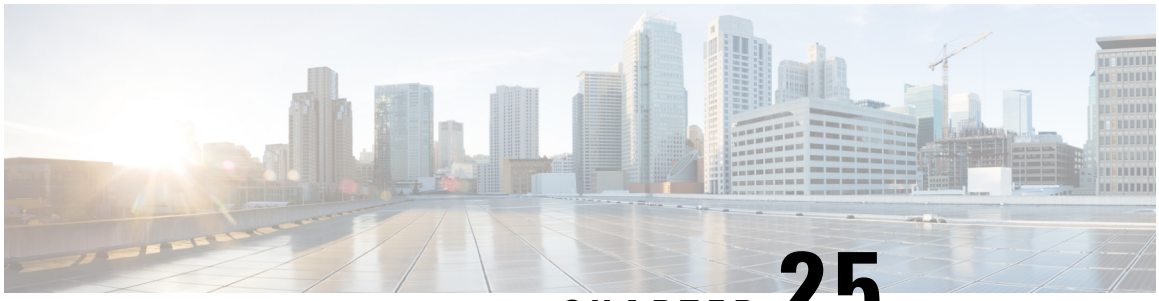
This table provides release and related information for the features explained in this module.

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 Amsterdam 17.2.1	Hierarchical VPLS with MPLS Access	Configuring VPLS requires a full mesh of tunnel LSPs between all the PE devices that participate in the VPLS. With full-mesh configuration, signaling overhead and packet replication requirements for each provisioned VC on a PE device are high. Configuring Hierarchical VPLS with MPLS Access reduces signaling overhead and packet replication between devices.s
Cisco IOS XE Cupertino 17.7.1	Hierarchical VPLS with MPLS Access	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 25

Configuring VPLS: Routed Pseudowire IRB for IPv4 Unicast

The VPLS: Routed Pseudowire IRB for IPv4 Unicast feature allows a switch interface to route traffic instead of using a router.

Restrictions for Configuring VPLS: Routed Pseudowire IRB for IPv4 Unicast

- This feature is not supported on a domain configured with multicast routing protocols.
- This feature is not supported for the IPv6 address family.
- VPLS over GRE is not supported with integrated routing and bridging (IRB).

Information About VPLS: Routed Pseudowire IRB for IPv4 Unicast

The following sections provide information about VPLS: Routed Pseudowire IRB for IPv4 Unicast.

About VPLS: Routed Pseudowire IRB for IPv4 Unicast

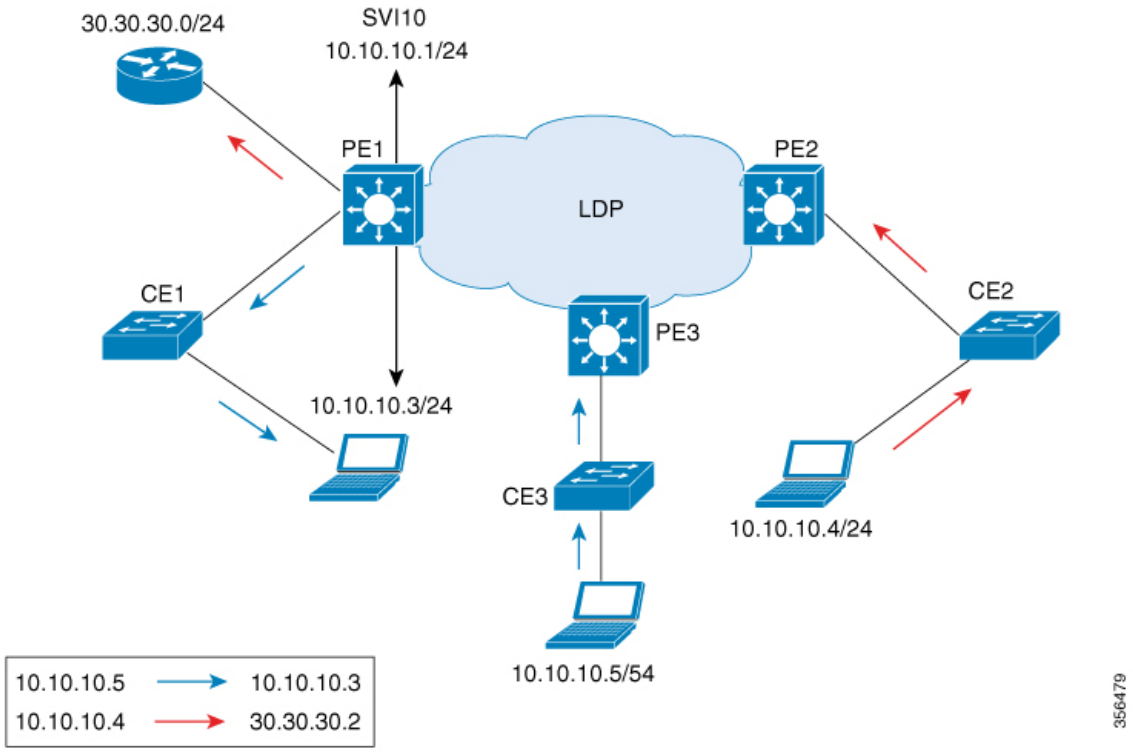
The VPLS: Routed Pseudowire IRB for IPv4 Unicast feature allows a Virtual Private LAN Services (VPLS) multipoint provider edge (PE) device interface to route the Layer 3 traffic along with switch the Layer 2 frames for pseudowire (PW) connections between PE devices. Note that the ability to route frames between interfaces does not affect the termination of a PW into the Layer 3 network (VPN or global) on the same device, or to tunnel Layer 3 frames over a Layer 2 tunnel (VPLS).

Centralized Integrated Routing and Bridging

In centralized Integrated Routing and Bridging (IRB), only one interface on a PE device is configured with IRB in the domain. All the host devices that are connected to PE devices are configured with this IRB interface IP address as the gateway.

The following figure shows a domain configured with centralized IRB. The figure shows that IRB is configured on the PE device (PE1) interface. All the hosts that are connected to the customer edge (CE) devices (CE1, CE2, and CE3), are configured with the IRB interface IP address (10.10.10.1) as the gateway. In this scenario, only those packets that are destined for the Layer 3 router (30.30.30.0/24) undergo Layer 3 packet rewrite because these interfaces or routers are reachable from the PE1 device. All the hosts communicate only in Layer 2 because they are part of the same bridge domain (10.10.10.x).

Figure 31: Centralized IRB



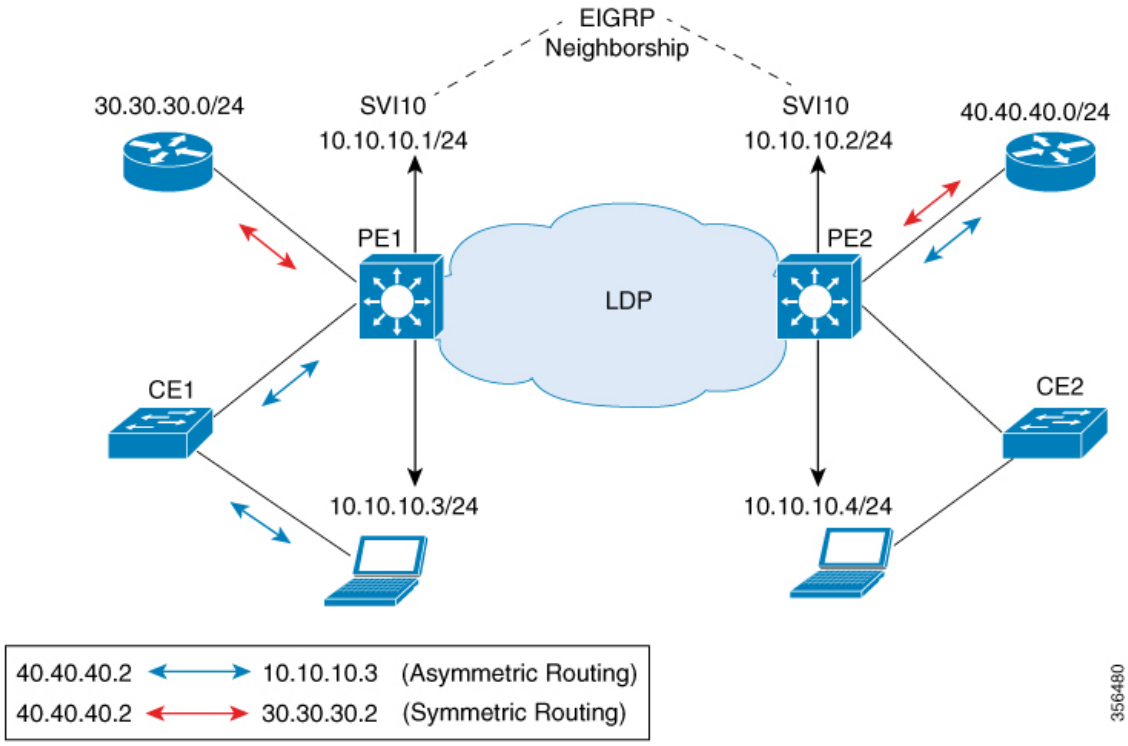
Distributed Integrated Routing and Bridging

In distributed IRB, all the interfaces across all the PE devices are configured with IRB in the domain. The routing protocols enabled on the PE devices allow routes to be learnt between PE devices.

The following figure shows a domain that is configured with distributed IRB. Enhanced Interior Gateway Routing Protocol (EIGRP) is configured on the interfaces of the PE devices (PE1 and PE2), which allows routers (30.30.30.0/24 and 40.40.40.0/24) to exchange routes. Hosts connected to the CE devices are configured with the local IRB interface IP address as the gateway. For example, host 10.10.10.3 is configured with IRB interface IP address 10.10.10.1 as the gateway, and host 10.10.10.4 is configured with IRB interface IP address 10.10.10.2 as the gateway. In this scenario, if the incoming traffic is through a switch virtual interface (SVI),

the outgoing traffic can also be reached by SVI through the MPLS network because the relationship is formed across IRB interfaces under the same bridge domain (10.10.10.x).

Figure 32: Distributed IRB



In the above diagram, where traffic is incoming on PE1 destined for a router interface reachable through PE2, routing takes place on egress of the PE (that is, PE2) based on the gateway configuration. In such a scenario, the packet reaching PE2 always has the source MAC as host MAC, and not the gateway MAC (which ages out after aging time). If the gateway MAC ages out, flooding occurs in the reverse direction traffic. Therefore, we recommend that in case of asymmetric routing, you configure an ARP timeout on the IRB interface that is lower than the MAC aging time so that flooding does not occur across PEs in the VPLS domain.

In this scenario (where traffic is incoming from CE1), both ingress and egress interfaces point to the SVI in the forwarding pipeline of PE1. Although this is expected, it generates ICMP redirect messages. Therefore, we recommend that you configure **no ip redirects** command on the SVI in interface configuration mode so that ICMP redirect messages are not generated in case of distributed IRB.

Features Supported with VPLS: Routed Pseudowire IRB for IPv4 Unicast

The following are the features that are supported on an interface that is configured with the VPLS: Routed Pseudowire IRB for IPv4 Unicast feature:

- IPv4 unicast routing protocols
- Virtual routing and forwarding (VRF)
- DHCP relay
- Address Resolution Protocol (ARP) timeout

- Blocking of Internet Control Message Protocol (ICMP) redirect messages

Configuring VPLS: Routed Pseudowire IRB for IPv4 Unicast

To configure VPLS: Routed Pseudowire IRB for IPv4 Unicast, 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 100`	Configures a VLAN interface and enters interface configuration mode
Step 4	**xconnect vfi** *vfi-name* **Example:** `Device(config-if)# xconnect vfi VFI100`	Specifies the Layer 2 VFI that you are binding to the VLAN port.
Step 5	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 10.10.10.1 255.255.255.0`	Assigns the IP address to the interface.

Example: Configuring Distributed IRB

The following example shows how to configure distributed IRB:

```
Device> enable
Device# configure terminal
Device(config)# template type pseudowire VPLS
Device(config-template)# encapsulation mpls
Device(config-template)# l2vpn vfi context VPLS
Device(config-template)# vpn id 10
Device(config-template)# member pseudowire1
Device(config-if)# end

Device(config)# interface pseudowire1
Device(config-if)# source template type pseudowire VPLS
Device(config-if)# encapsulation mpls
```

```
Device(config-if)# signaling protocol ldp
Device(config-if)# neighbor 10.10.10.10 10
Device(config-if)# end

Device(config)# interface Vlan10
Device(config-if)# ip address 10.10.10.1 255.255.255.0
Device(config-if)# no ip redirects
Device(config-if)# member vfi VPLS
Device(config-if)# end
```

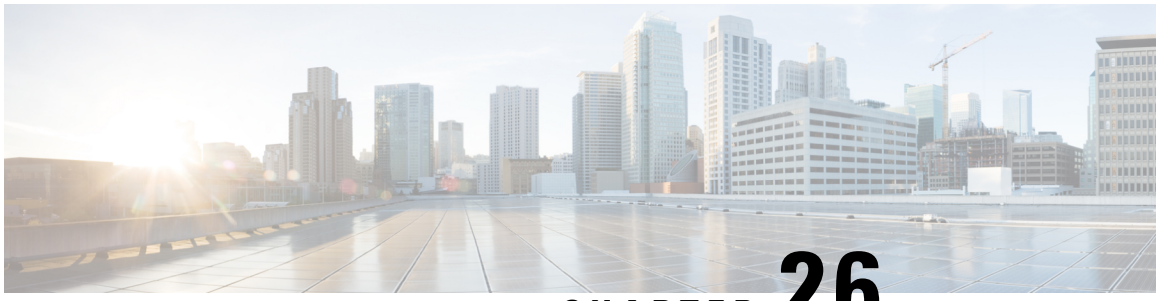
Feature History for Configuring VPLS: Routed Pseudowire IRB for IPv4 Unicast

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 Amsterdam 17.2.1	VPLS: Routed Pseudowire IRB for IPv4 Unicast	The VPLS: Routed Pseudowire IRB for IPv4 Unicast feature allows a switch interface to route traffic instead of using a router.
Cisco IOS XE Cupertino 17.7.1	VPLS: Routed Pseudowire IRB for IPv4 Unicast	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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/



CHAPTER 26

Configuring VPLS: Routed Pseudowire IRB for IPv6 Unicast

The VPLS: Routed Pseudowire IRB for IPv6 Unicast feature allows a switch interface to route traffic instead of using a router.

Restrictions for Configuring VPLS: Routed Pseudowire IRB for IPv6 Unicast

- This feature is not supported on a domain configured with multicast routing protocols.
- This feature is not supported for the IPv6 address family.
- VPLS over GRE is not supported with integrated routing and bridging (IRB).

Information About VPLS: Routed Pseudowire IRB for IPv6 Unicast

The following sections provide information about VPLS: Routed Pseudowire IRB for IPv6 Unicast.

About VPLS: Routed Pseudowire IRB for IPv6 Unicast

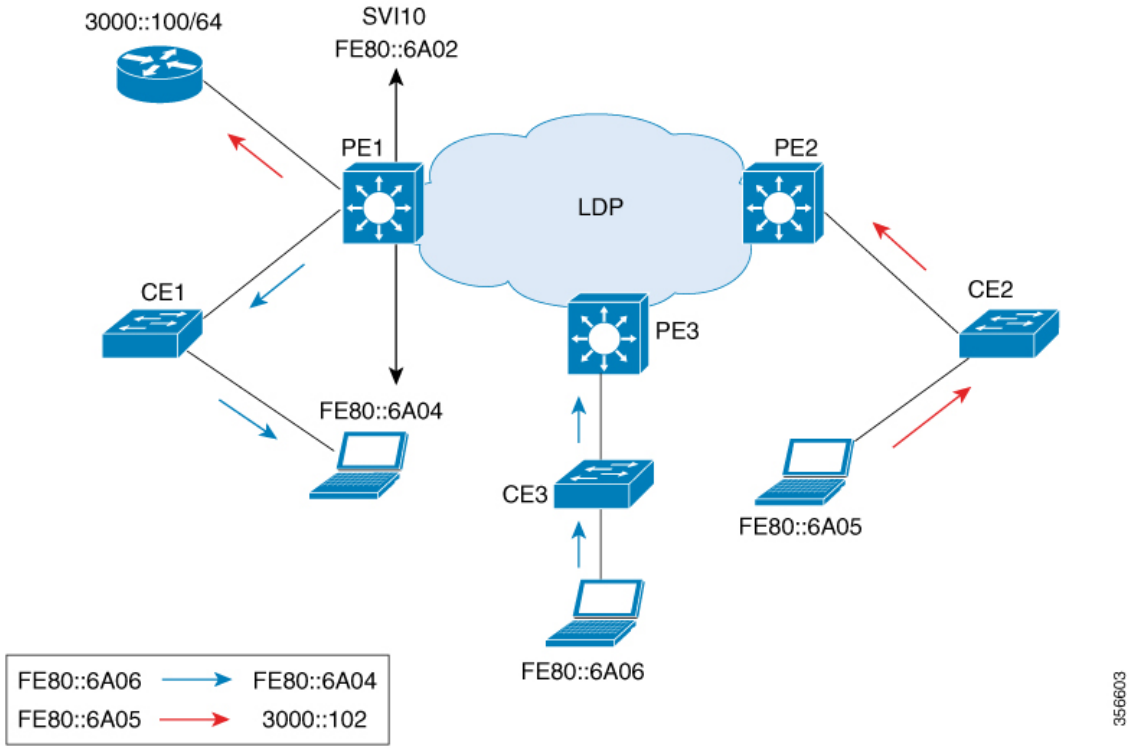
The VPLS: Routed Pseudowire IRB for IPv6 Unicast feature allows a Virtual Private LAN Services (VPLS) multipoint provider edge (PE) device interface to route the Layer 3 traffic along with switch the Layer 2 frames for pseudowire (PW) connections between PE devices. Note that the ability to route frames between interfaces does not affect the termination of a PW into the Layer 3 network (VPN or global) on the same device, or to tunnel Layer 3 frames over a Layer 2 tunnel (VPLS).

Centralized Integrated Routing and Bridging

In centralized Integrated Routing and Bridging (IRB), only one interface on a PE device is configured with IRB in the domain. All the host devices that are connected to PE devices are configured with this IRB interface IP address as the gateway.

The following figure shows a domain configured with centralized IRB. The figure shows that IRB is configured on the PE device (PE1) interface. All the hosts that are connected to the customer edge (CE) devices (CE1, CE2, and CE3), are configured with the IRB interface IPv6 address (FE80::6A02) as the gateway. In this scenario, only those packets that are destined for the Layer 3 router (3000::100/64) undergo Layer 3 packet rewrite because these interfaces or routers are reachable from the PE1 device. All the hosts communicate only in Layer 2 because they are part of the same bridge domain (FE80:6A0x)..

Figure 33: Centralized IRB



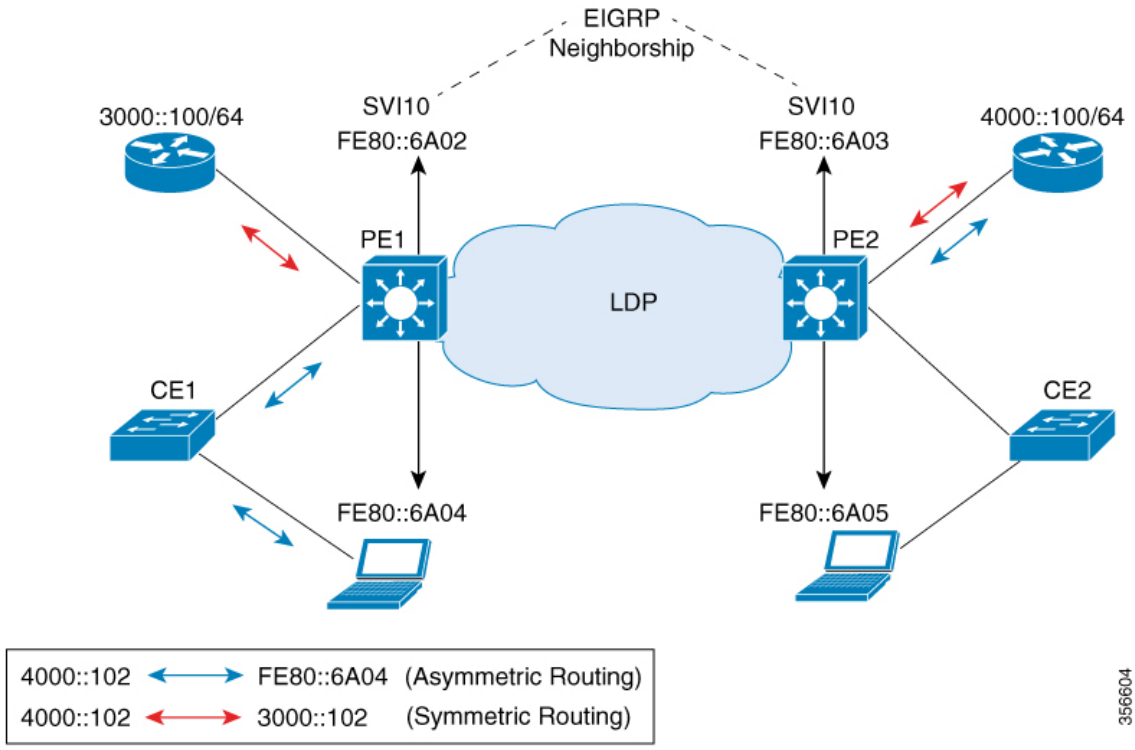
Distributed Integrated Routing and Bridging

In distributed IRB, all the interfaces across all the PE devices are configured with IRB in the domain. The routing protocols enabled on the PE devices allow routes to be learnt between PE devices.

The following figure shows a domain that is configured with distributed IRB. Enhanced Interior Gateway Routing Protocol (EIGRP) is configured on the interfaces of the PE devices (PE1 and PE2), which allows routers (3000::100/64 and 4000::100/64) to exchange routes. Hosts connected to the CE devices are configured with the local IRB interface IP address as the gateway. For example, host FE80::6A04 is configured with IRB interface IPv6 address FE80::6A02 as the gateway, and host FE80::6A05 is configured with IRB interface IPv6 address FE80::6A03 as the gateway. In this scenario, if the incoming traffic is through a switch virtual

interface (SVI), the outgoing traffic can also be reached by SVI through the MPLS network because the relationship is formed across IRB interfaces under the same bridge domain (FE80::6A0x).

Figure 34: Distributed IRB



In the above diagram, where traffic is incoming on PE1 destined for a router interface reachable through PE2, routing takes place on egress of the PE (that is, PE2) based on the gateway configuration. In such a scenario, the packet reaching PE2 always has the source MAC as host MAC, and not the gateway MAC (which ages out after aging time). If the gateway MAC ages out, flooding occurs in the reverse direction traffic. Therefore, we recommend that in case of asymmetric routing, you configure both the **ipv6 nd cache expire refresh** command and the **ipv6 nd cache expire** *timer* **refresh** command on the IRB interface, with the *timer* value lower than the MAC aging time so that flooding does not occur across PEs in the VPLS domain.

In this scenario (where traffic is incoming from CE1), both ingress and egress interfaces point to the SVI in the forwarding pipeline of PE1. Although this is expected, it generates ICMP redirect messages. Therefore, we recommend that you configure **no ip redirects** command on the SVI in interface configuration mode so that ICMP redirect messages are not generated in case of distributed IRB.

Features Supported with VPLS: Routed Pseudowire IRB for IPv6 Unicast

The following are the features that are supported on an interface that is configured with the VPLS: Routed Pseudowire IRB for IPv6 Unicast feature:

- IPv6 unicast routing protocols
- Virtual routing and forwarding (VRF)
- DHCP relay

- Address Resolution Protocol (ARP) timeout
- Blocking of Internet Control Message Protocol (ICMP) redirect messages

Configuring VPLS: Routed Pseudowire IRB for IPv6 Unicast

To configure the VPLS: Routed Pseudowire IRB for IPv6 Unicast feature, 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 100`	Configures a VLAN interface and enters interface configuration mode
Step 4	**xconnect vfi** *vfi-name* **Example:** `Device(config-if)# xconnect vfi VFI100`	Specifies the Layer 2 VFI that you are binding to the VLAN port.
Step 5	**ipv6 address** *ip-address* **Example:** `Device(config-if)# ipv6 address 4000::100/64`	Assigns the IPv6 address to the interface.

Configuration Example: Distributed IRB

The following example shows how to configure distributed IRB:

```
Device> enable
Device# configure terminal
Device(config)# template type pseudowire VPLS
Device(config-template)# encapsulation mpls
Device(config-template)# l2vpn vfi context VPLS
Device(config-template)# vpn id 10
Device(config-template)# member pseudowire1
Device(config-if)# end

Device(config)# interface pseudowire1
```

```
Device(config-if)# source template type pseudowire VPLS
Device(config-if)# encapsulation mpls
Device(config-if)# signaling protocol ldp
Device(config-if)# neighbor 3000::102
Device(config-if)# end

Device(config)# interface Vlan10
Device(config-if)# ipv6 address 4000::100/64
Device(config-if)# no ip redirects
Device(config-if)# member vfi VPLS
Device(config-if)# end
```

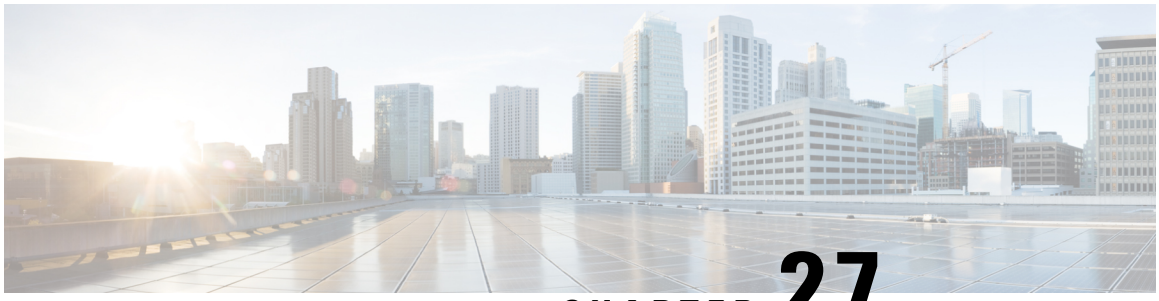
Feature History for Configuring VPLS: Routed Pseudowire IRB for IPv6 Unicast

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 Name	Feature Information
Cisco IOS XE Amsterdam 17.3.x	VPLS: Routed Pseudowire IRB for IPv6 Unicast	The VPLS: Routed Pseudowire IRB for IPv6 Unicast feature allows a switch interface to route traffic instead of using a router. Support for this feature was introduced on the Cisco Catalyst 9400 Series Switches.
Cisco IOS XE Cupertino 17.7.1	VPLS: Routed Pseudowire IRB for IPv6 Unicast	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release

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 27

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

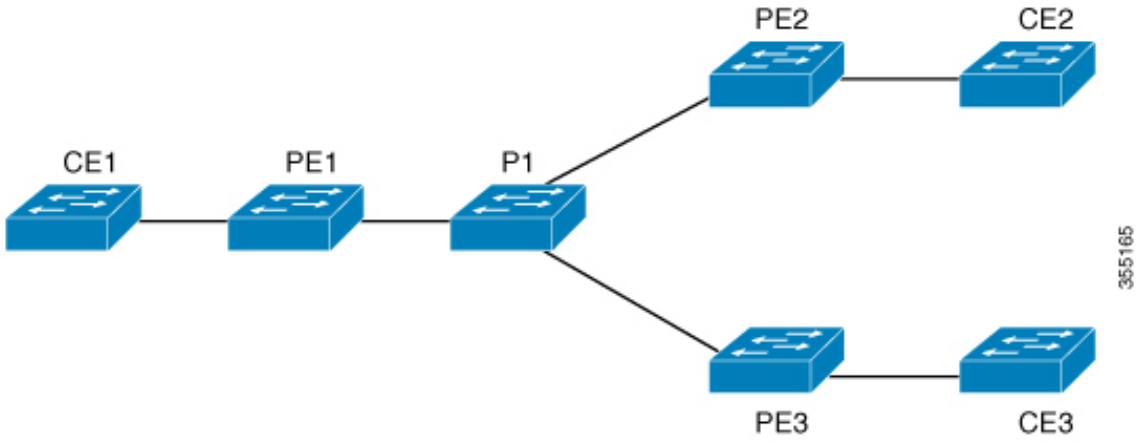
Routing policies for a peer include all configurations that may impact inbound or outbound routing table updates. The MPLS VPN Route Target Rewrite feature can influence routing table updates by allowing the replacement of route targets on inbound and outbound Border Gateway Protocol (BGP) updates. Route 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 35: 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
```

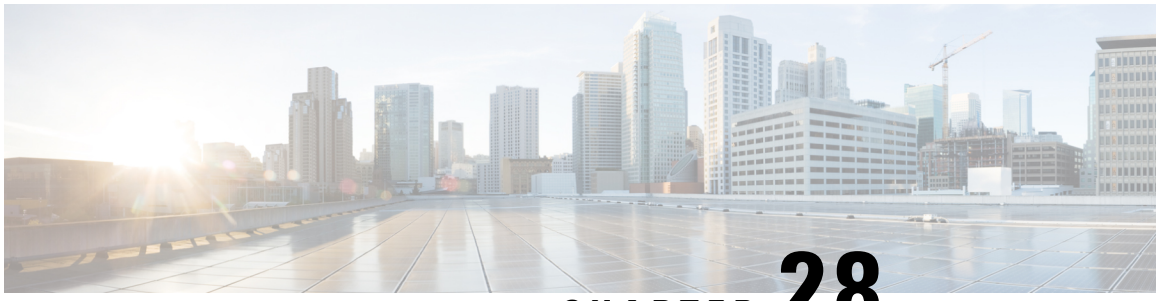
Feature History for MPLS VPN Route Target Rewrite

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 Fuji 16.9.1	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.
Cisco IOS XE Cupertino 17.7.1	MPLS VPN Route Target Rewrite	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 28

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:

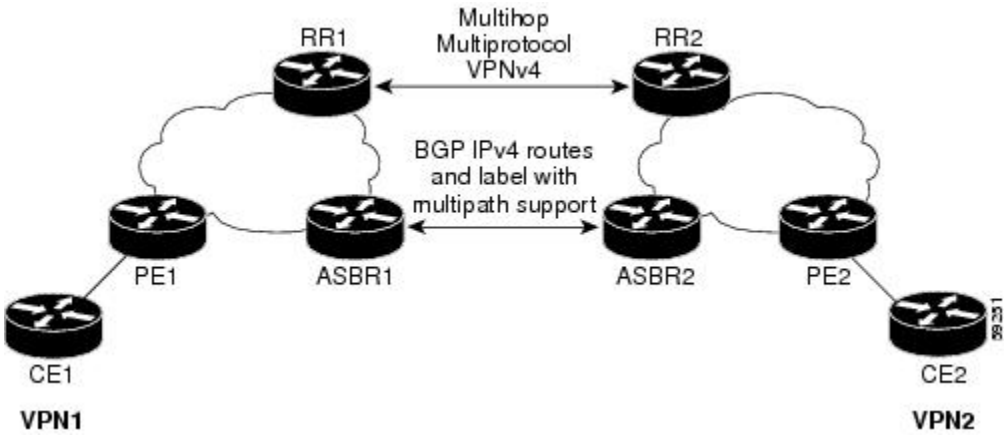
MPLS VPN Inter-AS IPv4 BGP Label Distribution Overview

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 36: 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:

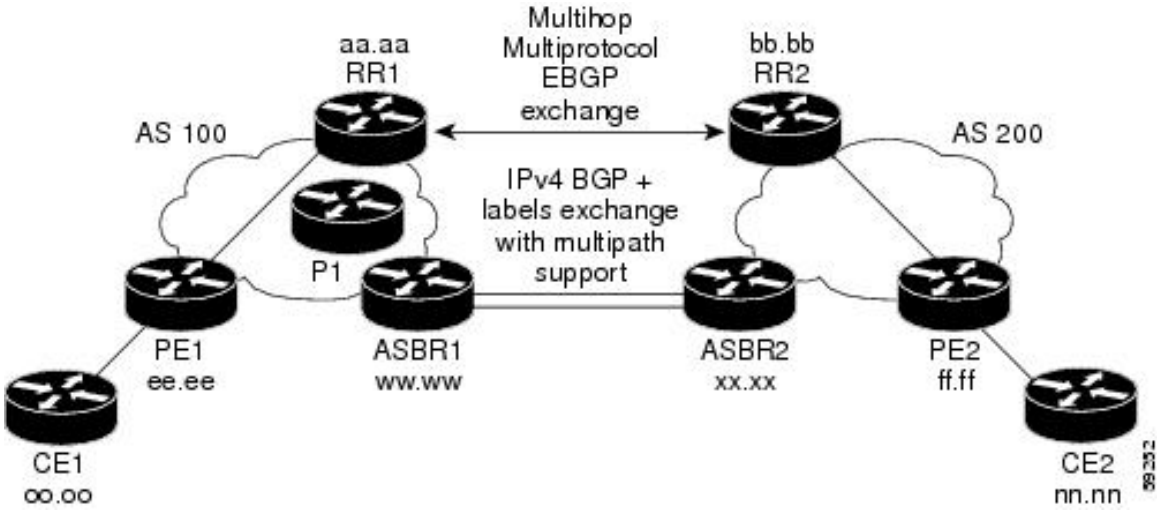
- For a router distributing MPLS labels, you can specify which routes are distributed with an MPLS label.
- For a router receiving MPLS labels, you can specify which routes are accepted and installed in the BGP table.

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 37: 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.

	Command or Action	Purpose
		Note The valid values of the **maximum-paths** command range from 1 to 32. However, the maximum value that can be configured is 2.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 209.165.201.2 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 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.

	Command or Action	Purpose
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 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.

	Command or Action	Purpose
		• 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.
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.

	Command or Action	Purpose
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* | *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.

	Command or Action	Purpose
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:** `Device(config-router-af)# exit-address-family`	Exits from the address family submode.
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.

	Command or Action	Purpose
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 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

	Command or Action	Purpose
		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.
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

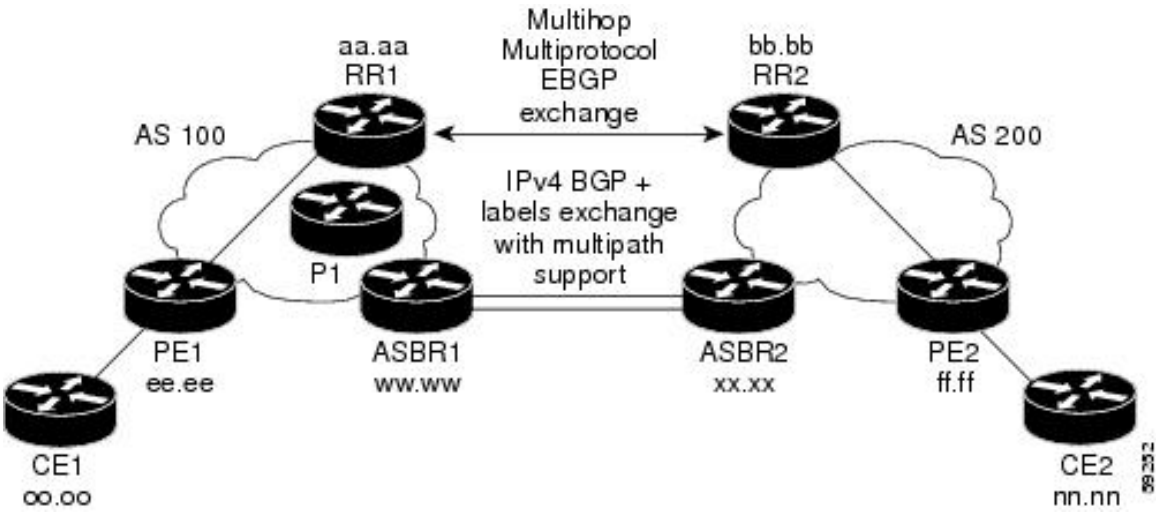
	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Device(config)# 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	**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:**	Applies a route map to incoming routes.

	Command or Action	Purpose
	`Device(config-router-af)# neighbor 209.165.200.225 route-map OUT out`	• 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.
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 38: 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.

	Command or Action	Purpose
		• 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.
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**] [**neighbors**] [**path** [*line*]] [**peer-group**] [**quote-regexp**] [**regexp**] [**summary**] [**tags**] **Example:** `Device# show ip bgp vpnv4 vrf vpn1 209.165.201.1`	(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.
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:**	(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).

	Command or Action	Purpose
	`Device# show mpls forwarding-table`	
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**	(Optional) Displays the contents of the LFIB. • Use the **show mpls forwarding-table** command with the **vrf** keyword to check

	Command or Action	Purpose
	label [*-label*] | **interface** *interface* | **next-hop** *address* | **lsp-tunnel** [*tunnel-id*] }] [**detail**] **Example:** `Device# show mpls forwarding-table vrf vpn1 209.165.201.1`	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.

	Command or Action	Purpose
		• 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.

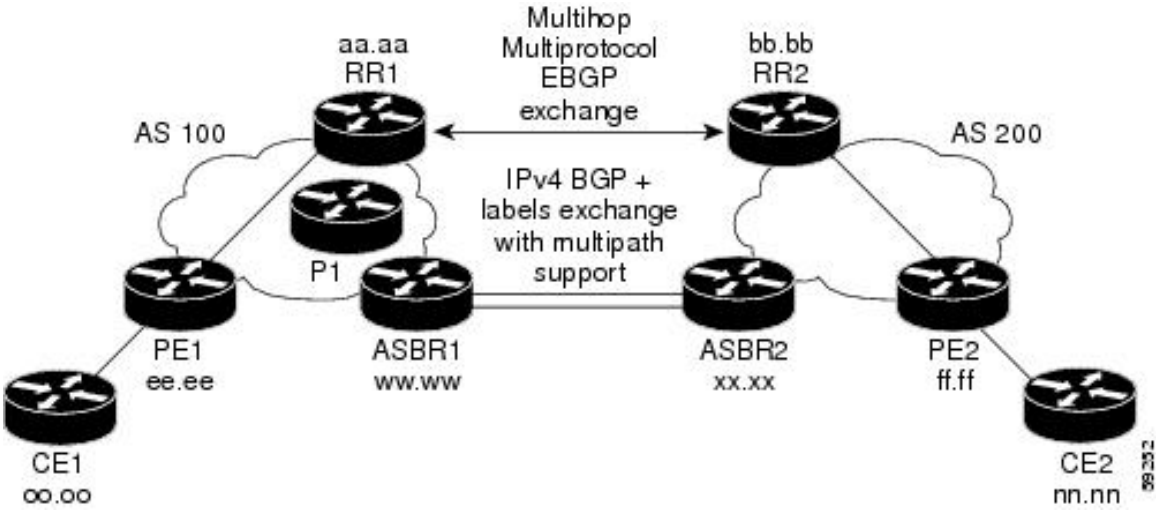
Configuration Examples for MPLS VPN Inter-AS IPv4 BGP Label Distribution

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 39: 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.

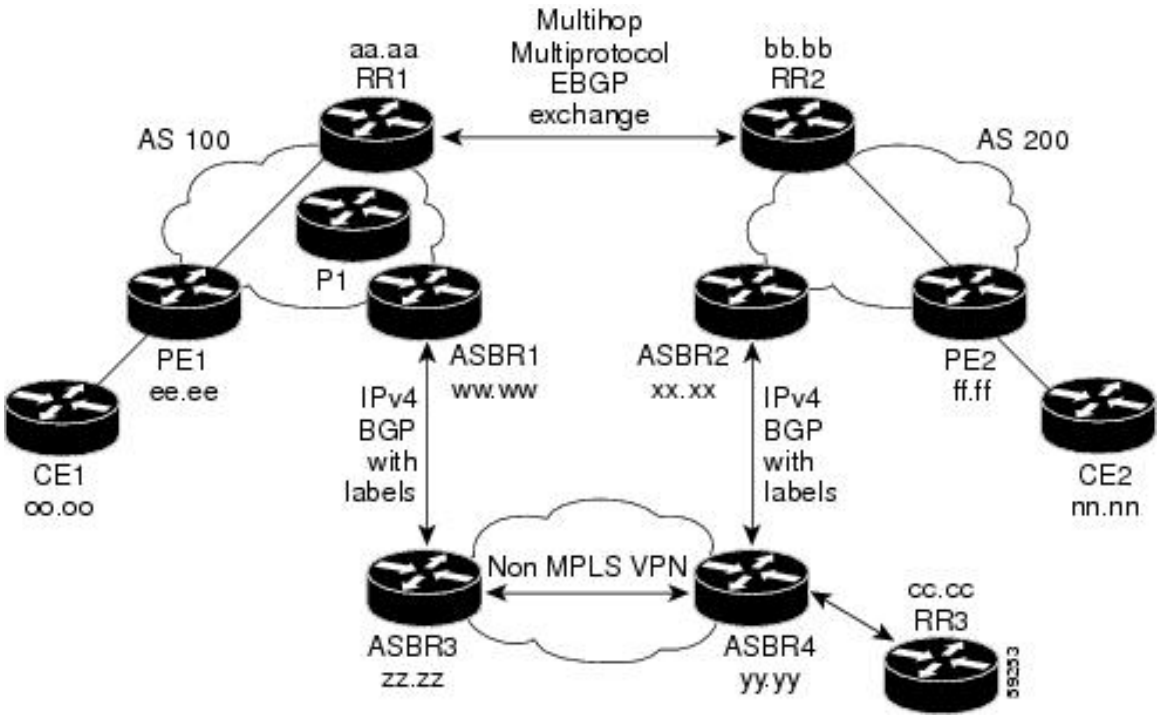
```
ip subnet-zero
 ip cef
 !
 mpls label protocol tdp
 !
 interface Loopback0
  ip address 209.165.200.226 255.255.255.255
  no ip directed-broadcast
 !
 interface Ethernet1/0
  ip address 209.165.201.2 255.0.0.0
  no ip directed-broadcast
  no ip mroute-cache
 !
 interface Ethernet1/2
  ip address 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 40: 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
```

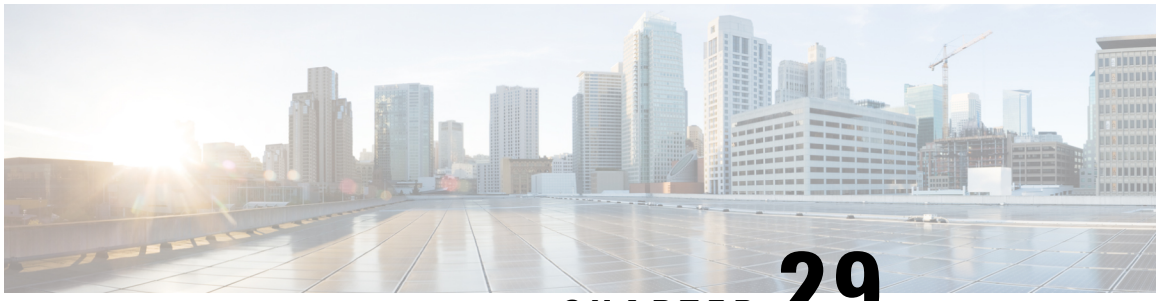
Feature History for Configuring MPLS VPN Inter-AS IPv4 BGP Label Distribution

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 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.
Cisco IOS XE Cupertino 17.7.1	MPLS VPN Inter-AS IPv4 BGP Label Distribution	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 29

Configuring Seamless MPLS

Information about Seamless MPLS

The following sections provide information about Seamless MPLS.

Overview of Seamless MPLS

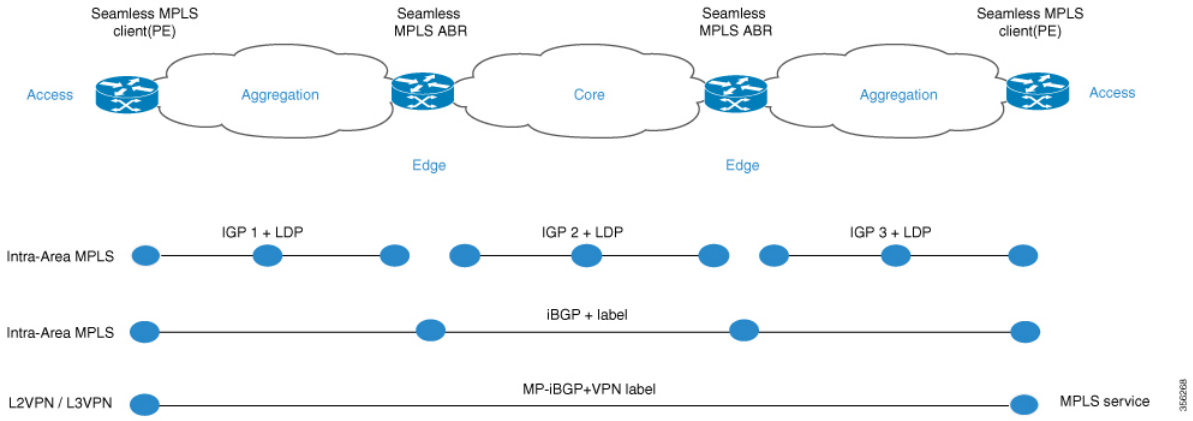
Seamless MPLS provides a highly flexible and scalable architecture to integrate multiple networks into a single MPLS domain. It is based on existing and well known protocols.

A large MPLS network can have several types of platforms and services in different parts of the network. Such a network would usually be divided into areas such as a core area and aggregation areas, and each of these areas have different Interior Gateway Protocols (IGPs). The IGP prefixes from one area cannot be distributed to another area. If the IGP prefixes cannot be distributed, then end-to-end Label-Switched-Paths (LSP) cannot be established. This affects the scalability of the network.

Seamless MPLS introduces greater scalability by establishing end-to-end LSPs. Seamless MPLS uses the Border Gateway Protocol (BGP) instead of IGP to forward the loopback prefixes of the Provider Edge (PE) routers. BGP distributes the prefixes end-to-end. This eliminates the need to install IGP prefixes of one domain in another domain.

Seamless MPLS introduces separation of the service and transport planes and provides end to end service independent transport. It removes the need for service specific configurations in network transport nodes.

Architecture for Seamless MPLS



The figure shows a network with three different areas: one core and two aggregation areas on the side. Each area runs its own IGP, with no redistribution between them on the Area Border Router (ABR). Use of BGP is needed in order to provide an end-to-end MPLS LSP. BGP advertises the loopbacks of the PE routers with a label across the whole domain, and provides an end-to-end LSP. BGP is deployed between the PEs and ABRs.

Seamless MPLs uses BGP to provide an end-to-end MPLS LSP. BGP is deployed between the PEs and the ABRs. BGP sends the IPv4 prefix and label. BGP advertises the loopbacks of the PE routers with a label across the whole domain and provides an end-to-end LSP.

When using IGP in the network, the next-hop address of the prefixes is the loopback prefix of the PE routers. This prefix is not known to the IGP being used in other parts of the network. The next hop address cannot be used to recurse to an IGP prefix. To avoid this the prefixes are carried in BGP. The ABRs are configured as Route Reflectors (RR). And the RRs are configured to set the next hop to self even for the reflected iBGP prefixes.

There are two possible scenarios.

- The ABR does not set the next hop to self for the prefixes advertised (reflected by BGP) by the ABR into the aggregation part of the network. The ABR needs to redistribute the loopback prefixes of the ABRs from the core IGP into the aggregation IGP. Only the ABR loopback prefixes (from the core) need to be advertised into the aggregation part, not the loopback prefixes from the PE routers from the remote aggregation parts.
- The ABR sets the next hop to self for the prefixes advertised (reflected by BGP) by the ABR into the aggregation part. Because of this, the ABR does not need to redistribute the loopback prefixes of the ABRs from the core IGP into the aggregation IGP.

In both scenarios, the ABR sets the next hop to self for the prefixes advertised (reflected by BGP) by the ABR from the aggregation part of the network into the core part.

How to configure Seamless MPLs

The following sections provide information on how to configure Seamless MPLS.

Configuring Seamless MPLS on the PE Router

The following steps can be used to configure Seamless MPLS on the PE Router

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 loopback** *slot/port* **Example:** `Device(config-if)# interface Loopback0`	Configures a Loopback interface and enters interface configuration mode.
Step 4	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)ip address 10.100.1.4 255.255.255.255`	Enters the IP address for the interface.
Step 5	**interface ethernet** *slot/port* **Example:** `Device(config-if)# interface Ethernet1/0`	Configures an Ethernet interface and enters interface configuration mode.
Step 6	**no ip address** **Example:** `Device(config-if)# no ip address`	Removes an IP address definition.
Step 7	**xconnect** *peer-ip-address vcid* **encapsulation mpls** **Example:** `Device(config-if)# xconnect 10.100.1.5 100 encapsulation mpls`	Specifies MPLS as the tunneling method to encapsulate.
Step 8	**router ospf** *process-id* **Example:** `Device(config)# router ospf 2`	Configures the OSPF routing process.
Step 9	**network** *ip-address wild-mask* **area** *area-id* **Example:**	Defines the interfaces on which OSPF runs and defines the area ID for those interfaces.

	Command or Action	Purpose
	`Device(config-router)# network 10.2.0.0 0.0.255.255 area 0`	
Step 10	**network** *ip-address wild-mask* **area** *area-id* **Example:** `Device(config-router)# network 10.100.1.4 0.0.0.0 area 0`	Defines the interfaces on which OSPF runs and defines the area ID for those interfaces.
Step 11	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1`	Configures the BGP routing process.
Step 12	**bgp log neighbor changes** **Example:** `Device(config-router)# bgp log neighbor changes`	Enables logging of BGP neighbor resets.
Step 13	**address-family ipv4** **Example:** `Device(config-router)# address-family ipv4`	Enters address family configuration mode.
Step 14	**network** *network-number* **mask** *network-mask* **Example:** `Device(config-router-af)# network 10.100.1.4 mask 255.255.255.255`	Specifies the networks to be advertised by BGP and multiprotocol BGP routing processes.
Step 15	**no bgp default ipv4 unicast** **Example:** `Device(config-router-af)# no bgp default ipv4 unicast`	Disables default IPv4 unicast address family for peering session establishment
Step 16	**no bgp default route-target filter** **Example:** `Device(config-router-af)# no bgp default route-target filter`	Disables automatic BGP route-target community filtering.
Step 17	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-router-af)# neighbor 10.100.1.1 remote-as 1`	Adds an entry to the BGP or multiprotocol BGP neighbor table.
Step 18	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router-af)# neighbor 10.100.1.1 update-source Loopback0`	Allows BGP sessions to use any operational interface for TCP connections.

	Command or Action	Purpose
Step 19	**neighbor** *ip-address* **send-label** **Example:** `Device(config-router-af)# neighbor 10.100.1.1 send-label`	Enables a BGP router to send MPLS labels with BGP routes to a neighboring BGP router.

Configuring Seamless MPLS on the Route Reflector

The following steps can be used to configure Seamless MPLS on the Route Reflector.

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 loopback** *slot/port* **Example:** `Device(config-if)# interface Loopback0`	Configures a Loopback interface and enters interface configuration mode.
Step 4	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.100.1.1 255.255.255.255`	Enters the IP address for the interface.
Step 5	**router ospf** *process-id* **Example:** `Device(config)# router ospf 1`	Configures the OSPF routing process.
Step 6	**network** *ip-address wild-mask* **area** *area-id* **Example:** `Device(config-router)# network 10.1.0.0 0.0.255.255 area 0`	Defines the interfaces on which OSPF runs and defines the area ID for those interfaces.
Step 7	**network** *ip-address wild-mask* **area** *area-id* **Example:** `Device(config-router)# 10.100.1.1 0.0.0.0 area 0`	Defines the interfaces on which OSPF runs and defines the area ID for those interfaces.

	Command or Action	Purpose
Step 8	**exit** **Example:** `Device(config-router)#exit`	Exits the configuration mode.
Step 9	**router ospf** *process-id* **Example:** `Device(config)# router ospf 2`	Configures the OSPF routing process.
Step 10	**redistribute ospf** *instance-tag* **route-map** *map-name* **Example:** `Device(config-router)# redistribute ospf 1 subnets match internal route-map ospf1-into-ospf2`	Injects routes from one routing domain into OSPF.
Step 11	**network** *ip-address wild-mask* **area** *area-id* **Example:** `Device(config-router)# network 10.2.0.0 0.0.255.255 area 0`	Defines the interfaces on which OSPF runs and defines the area ID for those interfaces.
Step 12	**exit** **Example:** `Device(config-router)#exit`	Exits the configuration mode.
Step 13	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 1`	Configures the BGP routing process.
Step 14	**bgp log neighbor changes** **Example:** `Device(config-router)# bgp log neighbor changes`	Enables logging of BGP neighbor resets.
Step 15	**address-family ipv4** **Example:** `Device(config-router)# address family ipv4`	Enters address family configuration mode.
Step 16	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-route-af)# neighbor 10.100.1.2 remote-as 1`	Adds an entry to the BGP or multiprotocol BGP neighbor table.

	Command or Action	Purpose
Step 17	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router-af)# neighbor 10.100.1.2 update-source Loopback0`	Allows BGP sessions to use any operational interface for TCP connections.
Step 18	**neighbor** *ip-address* **next-hop-self all** **Example:** `Device(config-router-af)# neighbor 10.100.1.2 next-hop-self all`	Configures a router as the next hop for a BGP-speaking neighbor or peer group.
Step 19	**neighbor** *ip-address* **send-label** **Example:** `Device(config-router-af)# neighbor 10.100.1.2  send-label`	Enables a BGP router to send MPLS labels with BGP routes to a neighboring BGP router.
Step 20	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-router-af)# neighbor 10.100.1.4 remote-as 1`	Adds an entry to the BGP or multiprotocol BGP neighbor table.
Step 21	**neighbor** *ip-address* **update-source** *interface-type interface-number* **Example:** `Device(config-router-af)# neighbor 10.100.1.4 update-source Loopback0`	Allows BGP sessions to use any operational interface for TCP connections.
Step 22	**neighbor** *ip-address* **route-reflector-client** **Example:** `Device(config_router-af)# neighbor 10.100.1.4 route-reflector-client`	Configures the router as a BGP route reflector and configure the specified neighbor as its client.
Step 23	**neighbor** *ip-address* **next-hop-self all** **Example:** `Device(config-router-af)# neighbor 10.100.1.4 next-hop-self all`	Configures a router as the next hop for a BGP-speaking neighbor or peer group.
Step 24	**neighbor** *ip-address* **send-label** **Example:** `Device(config-router-af)# neighbor 10.100.1.4 send-label`	Enables a BGP router to send MPLS labels with BGP routes to a neighboring BGP router.
Step 25	**exit** **Example:** `Device(config-router)#exit`	Exits the configuration mode.

	Command or Action	Purpose
Step 26	**ip prefix-list** *name* **seq** *number* **permit** *prefix* **Example:** `Device(config)# ip prefix-list prefix-list-ospf1-into-ospf2 seq 5 permit 10.100.1.1/32`	Creates a prefix list to match IP packets or routes against.
Step 27	**route-map** *name* **permit** *sequence-number* **Example:** `Device(config)# route-map ospf1-into-ospf2 permit 10`	Creates the route map entry. Enters route-map configuration mode.
Step 28	**match ip address prefix-list** *prefix-list-name* **Example:** `Device(config-route-map)# match ip address prefix-list prefix-list-ospf1-into-ospf2`	Distributes routes that have a destination IP network number address that is permitted by a prefix list.

Configuration Examples for Seamless MPLS

The following sections provide examples for configuring Seamless MPLS.

Example: Configuring Seamless MPLS on PE Router 1

The following example shows how to configure Seamless MPLS on PE router 1.

```
Device(config-if)#interface Loopback0
 Device(config-if)#ip address 10.100.1.4 255.255.255.255
!
Device(config-if)# interface Ethernet1/0
Device(config-if)# no ip address
Device(config-if)# xconnect 10.100.1.5 100 encapsulation mpls
!
Device(config)# router ospf 2
Device(config-router)# network 10.2.0.0 0.0.255.255 area 0
Device(config-router)# network 10.100.1.4 0.0.0.0 area 0
!
Device(config)#router bgp 1
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# address family ipv4
Device(config-router-af)# network 10.100.1.4 mask 255.255.255.255
Device(config-router-af)# no bgp default ipv4 unicast
Device(config-router-af)# no bgp default route-target filter
Device(config-router-af)# neighbor 10.100.1.1 remote-as 1
Device(config-router-af)# neighbor 10.100.1.1 update-source Loopback0
Device(config-router-af)# neighbor 10.100.1.1 send-label
```

Example: Configuring Seamless MPLS on Route Reflector 1

The following examples shows how to configure Seamless MPLS on route reflector 1.

```
Device(cofig-if)# interface Loopback0
Device(cofig-if)# ip address 10.100.1.1 255.255.255.255
Device(config)# router ospf 1
Device(config-router)# network 10.1.0.0 0.0.255.255 area 0
Device(config-router)# network 10.100.1.1 0.0.0.0 area 0
!
Device(config)# router ospf 2
Device(config-router)# redistribute ospf 1 subnets match internal route-map ospf1-into-ospf2
Device(config-router)# network 10.2.0.0 0.0.255.255 area 0
!
Device(config)# router bgp 1
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# address family ipv4
Device(config-router-af)# neighbor 10.100.1.2 remote-as 1
Device(config-router-af)# neighbor 10.100.1.2 update-source Loopback0
Device(config-router-af)# neighbor 10.100.1.2 next-hop-self all
Device(config-router-af)# neighbor 10.100.1.2 send-label
Device(config-router-af)# neighbor 10.100.1.4 remote-as 1
Device(config-router-af)# neighbor 10.100.1.4 update-source Loopback0
Device(config-router-af)# neighbor 10.100.1.4 route-reflector-client
Device(config-router-af)# neighbor 10.100.1.4 next-hop-self all
Device(config-router-af)# neighbor 10.100.1.4 send-label

Device(config)# ip prefix-list prefix-list-ospf1-into-ospf2 seq 5 permit 10.100.1.1/32

Device(config)# route-map ospf1-into-ospf2 permit 10
Device(conifg-route-mao)# match ip address prefix-list prefix-list-ospf1-into-ospf2
```

Example: Configuring Seamless MPLS on PE Router 2

The following example shows how to configure Seamless MPLS on PE router 2.

```
Device(config-if)#interface Loopback0
 Device(config-if)#ip address 10.100.1.5 255.255.255.255
!
Device(config-if)# interface Ethernet1/0
Device(config-if)# no ip address
Device(config-if)# xconnect 10.100.1.4 100 encapsulation mpls
!
Device(config)# router ospf 3
Device(config-router)# network 10.3.0.0 0.0.255.255 area 0
Device(config-router)# network 10.100.1.5 0.0.0.0 area 0
!
Device(config)#router bgp 1
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# address family ipv4
Device(config-router-af)# network 10.100.1.5 mask 255.255.255.255
Device(config-router-af)# no bgp default ipv4 unicast
Device(config-router-af)# no bgp default route-target filter
Device(config-router-af)# neighbor 10.100.1.2 remote-as 1
Device(config-router-af)# neighbor 10.100.1.2 update-source Loopback0
Device(config-router-af)# neighbor 10.100.1.2 send-label
```

Example: Configuring Seamless MPLS on Route Reflector 2

The following examples shows how to configure Seamless MPLS on route reflector 2.

```
Device(cofig-if)# interface Loopback0
Device(cofig-if)# ip address 10.100.1.2 255.255.255.255
Device(config)# router ospf 1
Device(config-router)# network 10.1.0.0 0.0.255.255 area 0
```

```
Device(config-router)# network 10.100.1.2 0.0.0.0 area 0
!
Device(config)# router ospf 3
Device(config-router)# redistribute ospf 1 subnets match internal route-map ospf1-into-ospf3
Device(config-router)# network 10.3.0.0 0.0.255.255 area 0
!
Device(config)# router bgp 1
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# address family ipv4
Device(config-router-af)# neighbor 10.100.1.1 remote-as 1
Device(config-router-af)# neighbor 10.100.1.1 update-source Loopback0
Device(config-router-af)# neighbor 10.100.1.1 next-hop-self all
Device(config-router-af)# neighbor 10.100.1.1  send-label
Device(config-router-af)# neighbor 10.100.1.5 remote-as 1
Device(config-router-af)# neighbor 10.100.1.5 update-source Loopback0
Device(config-router-af)# neighbor 10.100.1.5 route-reflector-client
Device(config-router-af)# neighbor 10.100.1.5 next-hop-self all
Device(config-router-af)# neighbor 10.100.1.5 send-label

Device(config)# ip prefix-list prefix-list-ospf1-into-ospf3 seq 5 permit 10.100.1.1/32

Device(config)# route-map ospf1-into-ospf3 permit 10
Device(conifg-route-mao)# match ip address prefix-list prefix-list-ospf1-into-ospf3
```

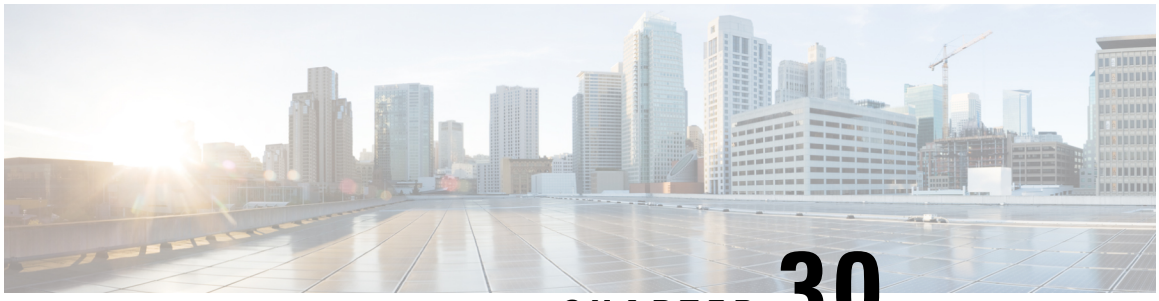
Feature History for Seamless MPLS

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 16.12.1	Seamless MPLS	Seamless MPLS provides a highly flexible and scalable architecture to integrate multiple networks into a single MPLS domain. It is based on existing and well known protocols.
Cisco IOS XE Cupertino 17.7.1	Seamless MPLS	This feature was implemented on supervisor modules C9400X-SUP-2 and C9400X-SUP-2XL, which were introduced in this release.

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 30

Troubleshooting Multiprotocol Label Switching

Overview

This chapter provides links to documents authored by Cisco subject matter experts (SMEs). They aim to help you resolve technical issues without requiring a support ticket. If these documents are unable to resolve your issue, we recommend visiting the applicable Cisco Community. There is a wealth of information and advice available from fellow Cisco customers who may have experienced this issue already and provided a solution. If you are not able to find a resolution on the Community, it may be best that you raise a support ticket at Cisco Support. In cases where a support ticket has to be raised, these documents provide guidance about the data that should be collected and added to the support ticket. Specify the support document you referred, and TAC can create an improvement request with the document owner.

Support Articles

The documents in this section were created using specific software and hardware listed in the Components Used section of each article. However, this does not mean that they are limited to what is listed in Components Used, and generally remain relevant for later versions of software and hardware. Note that there could be some changes in the software or hardware that can cause commands to stop working, the syntax to change, or GUIs and CLIs to look different from one release to another.

The following are the support articles associated with this technology:

Document	Description
Verify MPLS on Catalyst 9000 Switches	This document describes the how to configure and validate Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN) on Catalyst 9000 series switches.

Feedback Request

Your input helps. A key aspect to improving these support documents is customer feedback. Note that these documents are owned and maintained by multiple teams within Cisco. If you find an issue specific to the document (unclear, confusing, information missing, etc):

- Provide feedback using the **Feedback** button located at the right panel of the corresponding article. The document owner will be notified, and will either update the article, or flag it for removal.
- Include information regarding the section, area, or issue you had with the document and what could be improved. Provide as much detail as possible.

Disclaimer and Caution

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.