MPLS Configuration Guide

First Published: 2026-03-01

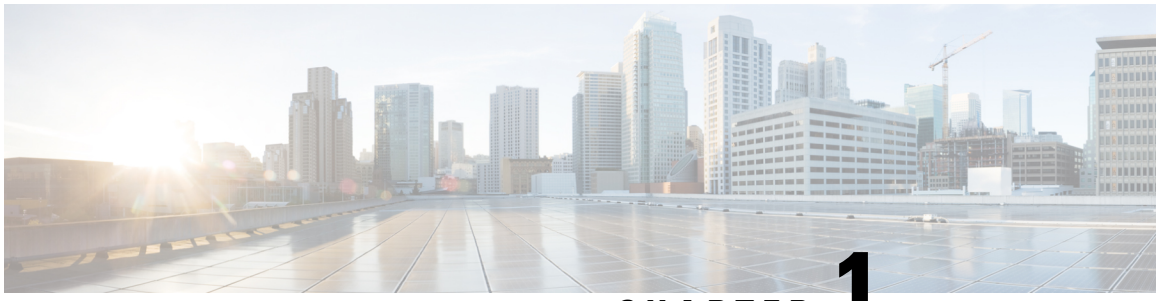
Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Read Me First

Only supported features are documented. To confirm or clarify all the supported features for a platform, go to Cisco Feature Navigator.

Read Me First

CONTENTS



CHAPTER 1

Introduction to MPLS

This module explains the basic concepts related to Multiprotocol Label Switching and describes how to configure it on Cisco Smart Switches.

Feature history for Multiprotocol Label Switching

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	Multiprotocol Label Switching	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

Multiprotocol Label Switching

A Multiprotocol Label Switching (MPLS) system is a high-performance packet forwarding technology that

- combines the performance and capabilities of Layer 2 (data link layer) switching with the scalability and flexibility of Layer 3 (network layer) routing
- optimizes transit by using labels instead of analyzing full headers at each switch, and
- supports all Layer 3 protocols, allowing for scalability far beyond traditional networks.

Label switching is a packet forwarding technique that

- analyzes the Layer 3 packet header once at the edge

- assigns a fixed-length label for forwarding, and
- enables rapid, simple forwarding decisions at each network node.

Multiple different headers can map to the same label if they share the same forwarding path, grouping them into a forwarding equivalence class.

How label switching differs from conventional Layer 3 forwarding

Label switching differs from conventional Layer 3 forwarding in these aspects.

Label switching	Conventional Layer 3 forwarding
Analyzes the Layer 3 header only once at the network edge	Analyzes the Layer 3 header independently at each switch as the packet traverses the network
Maps the Layer 3 header to a fixed-length, unstructured label	Uses information from the Layer 3 header, often the destination address, for a routing table lookup at every hop
The initial label assignment may consider more than just the Layer 3 header, for example, routing policy	Forwarding decisions are typically based only on the Layer 3 header fields of the packet
Many different headers can map to the same label if they follow the same path, forwarding equivalence class (FEC)	Each header is handled individually, requiring repeated analysis and lookup at each switch
After labeling, forwarding decisions at each hop are based on the label, not the original packet header	Every switch examines the Layer 3 header of the packet and performs a potentially complex routing table lookup to decide the next hop
Swaps labels at each MPLS switch, and refers to the MPLS forwarding table for the next hop, allowing fast and simple forwarding	Header analysis and routing table lookup can be complex and must be repeated at every hop, leading to slower forwarding

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 these 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)

Benefits of Multiprotocol Label Switching

Multiprotocol Label Switching offers these benefits:

- Enables the differentiation of services without requiring substantial changes to existing infrastructure
- Helps address rapid growth in network utilization

- Operates flexibly across any combination of Layer 2 technologies

Restrictions for Multiprotocol Label Switching

The support for Multiprotocol Label Switching is subjected to these restrictions:

- Does not support MPLS fragmentation
- Does not support MPLS maximum transmission unit (MTU) configuration. MPLS MTU value equals the IP MTU value of the port or switch, by default.
- Supports only the default mode—per-VRF MPLS label allocation mode. However, the devices can inter-operate with remote peers operating in the per-prefix mode.
- Does not support the **ip unnumbered** command in MPLS configuration
- You cannot enable MPLS LDP on a Virtual Routing and Forwarding (VRF) interface.

How label switching works

Summary

The key components involved in the process are:

- Ingress router: analyzes the Layer 3 packet header, assigns a label, and adds the label to the packet
- Label switching routers (LSRs): swap labels and forward packets based on local label forwarding tables
- Forwarding equivalence class: a set of packets treated identically for forwarding, represented by the same label

Label switching uses an ingress router to assign a fixed-length label to each packet, allowing label switching routers (LSRs) to forward packets quickly based on label lookups, all while grouping similar packets into forwarding equivalence classes.

Workflow

These stages describe how label switching works:

1. The ingress router examines the Layer 3 header of an incoming packet and determines its forwarding equivalence class.
2. The router assigns a label based on this analysis and attaches a label header to the packet.
3. As the labeled packet traverses the MPLS network, each LSR uses the label to look up forwarding information, swaps the label as needed, and forwards the packet—without re-examining the original header.

   Forwarding decisions at subsequent hops can use routing policy, not just header contents.
4. The process continues until the packet reaches its destination or exits the MPLS domain.

Result

Packets are forwarded efficiently through the network with minimal processing at each hop, resulting in higher performance and scalability compared to conventional Layer 3 forwarding.

Configure a switch for MPLS switching

Before you begin

Enable Cisco Express Forwarding (CEF) before you configure the switch for MPLS switching.

Procedure

Step 1 Enable Cisco Express Forwarding (CEF) on the switch.

Example:

```
Device#configure
Device(config)#ip cef distributed
```

Step 2 Configure the range of local labels available for use with MPLS applications on packet interfaces.

Example:

```
Device(config)#mpls label range 16 4096
```

Step 3 Specify the label distribution protocol for the platform.

Example:

```
Device(config)#mpls label protocol ldp
Device(config)#end
```

Step 4 Verify whether Cisco Express Forwarding configuration was successful.

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#
```

Configure a switch for MPLS forwarding

Before you begin

Enable forwarding of IPv4 packets before you configure the switch for MPLS forwarding.

Procedure

Step 1 Enable MPLS forwarding of IPv4 packets along routed physical interfaces (Gigabit Ethernet), Switch Virtual Interface (SVI), or port channels.

Example:

For Gigabit Ethernet interface:

```
Device#configure terminal
Device(config)#interface TenGigabitEthernet 1/0/1
Device(config-if)#mpls ip
```

Example:

For SVI:

```
Device(config)#iinterface vlan 1000
Device(config-if)#mpls ip
```

Step 2 Specify the label distribution protocol for the interface.

Example:

```
Device(config-if)#mpls label protocol ldp
Device(config-if)#end
```

Note
You cannot enable MPLS LDP on a Virtual Routing and Forwarding (VRF) interface.

Step 3 Verify whether the MPLS forwarding configuration on the switch was successful.

a) Verify the running configuration on the switch.

Example:

For Gigabit Ethernet interface:

```
Device# show running-config interface TenGigabitEthernet 1/0/1

Building configuration...

Current configuration : 307 bytes
!
interface TenGigabitEthernet1/0/1
 no switchport
 ip address xx.xx.x.x xxx.xxx.xxx.x
 mpls ip
 mpls label protocol ldp
end
```

For SVI:

```
Device# show running-config 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
```

b) Verify MPLS interface details.

Example:

For Gigabit Ethernet interface:

```
Device# show mpls interfaces detail
Interface TenGigabitEthernet 1/0/1:
        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 SVI:

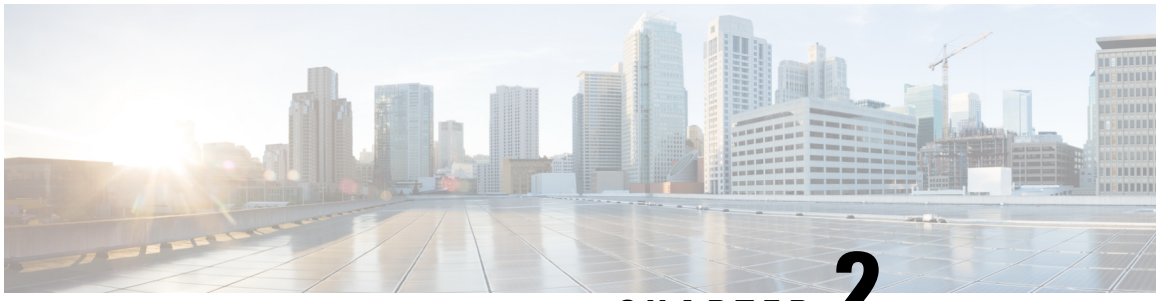
```
Device# show mpls interfaces detail
Interface Vlan100:
        Type Unknown
        IP labeling enabled (ldp) :
          Interface config
        LSP Tunnel labeling not enabled
        IP FRR labeling not enabled
        BGP labeling not enabled
        MPLS operational
        MTU = 1500
```

Step 4 Verify MPLS forwarding information on the switch.

Example:

```
For 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
```



CHAPTER 2

MPLS Static Labels

The MPLS static labels feature provides the means to configure the binding between a label and an IPv4 prefix statically. This module explains the basic concepts of MPLS static labels and how to configure it on Cisco Smart Switches.

Feature history for MPLS static labels

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS static labels	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS static labels

MPLS static label is a network configuration feature that

- provides the means to statically configure the binding between a label and an IPv4 prefix
- provides static and precise control over label assignments for packet forwarding, and
- serves as an alternative to protocol-based dynamic label distribution.

Generally, LSRs use label distribution protocols such as Label Distribution Protocol (LDP), Resource Reservation Protocol (RSVP), or Border Gateway Protocol (BGP) to dynamically learn and bind labels to network addresses. However, with MPLS static labels, the label-to-prefix association is set manually rather than learned through these protocols.

Once configured, the static label is installed in the Label Forwarding Information Base (LFIB) for use in label-switching packets.

Benefit of MPLS static labels

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.

Restrictions for MPLS static labels

The support for MPLS static labels is subjected to these restrictions:

- Does not support MPLS fragmentation
- Does not support MPLS maximum transmission unit (MTU) configuration
- Supports only the default mode—per-VRF MPLS label allocation mode. However, the devices can inter-operate with remote peers operating in the per-prefix mode.

Configure MPLS static label bindings

Procedure

Step 1 Specify a range of labels for static assignment.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls label range 200 100000 static 16 199
```

By default, no label is reserved for static assignment.

Note
You must reload the switch for the new MPLS label range configuration to take effect.

Step 2 Specify static binding of labels to IP prefix.

Example:

```
Device(config)# mpls static binding ipv4 10.0.0.0 255.0.0.0 55
Device(config)# end
```

Example:

You can configure input (local) and output (remote) labels for various prefixes as shown in this example:

```
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
```

Bindings specified are installed automatically in the MPLS forwarding table as routing demands.

Step 3 Verify MPLS static label bindings configuration on the switch.

a) Verify the MPLS static label range.

Example:

Before reload: The output shows that the new label range does not 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
```

Example:

After reload: The output 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
```

b) View the configured static label bindings.

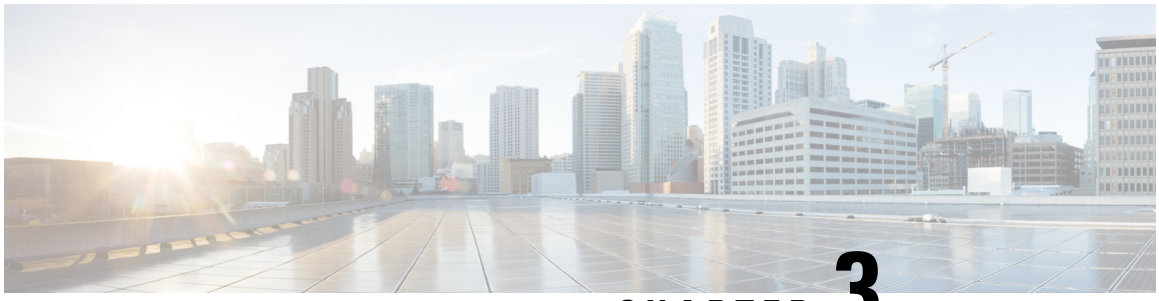
Example:

```
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
```

c) Check which static label bindings are currently in use for MPLS forwarding.

Example:

```
Device# show mpls forwarding-table
Local      Outgoing   Prefix           Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id     Switched      interface
20         No Label   IPv4 VRF[V]      0             aggregate/vrf1
24         explicit-n 5.5.5.5/32       0             Po18.19    118.118.1.2
28         20         2.2.2.2/32       0             Po18.19    118.118.1.2
525        No Label   IPv6 VRF[V]      1987078754344 aggregate/vrf1
29184      19         3.3.3.3/32       0             Po18.19    118.118.1.2
49024      explicit-n 109.1.1.0/24     0             Po18.19    118.118.1.2
49025      explicit-n 37.37.37.0/24    0             Po18.19    118.118.1.2
Device#
```



CHAPTER 3

MPLS VPN Route Target Rewrite

An MPLS VPN route target rewrite feature enables the replacement of route targets on inbound and outbound BGP updates to control which VPN routing and forwarding (VRF) instances and sites receive routes. This module explains the basic concepts of MPLS VPN route target rewrite and how to configure it on Cisco Smart Switches.

Feature history for MPLS VPN route target rewrite

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS VPN route target rewrite	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS VPN route target rewrite

MPLS VPN route target rewrite is an MPLS VPN routing control mechanism that

- influences routing table updates by modifying route targets
- enables the replacement of route targets on inbound and outbound Border Gateway Protocol (BGP) updates, and
- utilizes extended community attributes in BGP Virtual Private Network IP Version 4 (VPNv4) updates to control which VPN routing and forwarding (VRF) instances and sites receive routes.

Route targets (RTs) are BGP extended community attributes that

- are carried in BGP VPNv4 updates
- are used to identify which VRF instances and sites can receive specific VPN routes, and
- control the distribution of VPN routes between VRFs through export and import lists, acting as a filtering mechanism.

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** [*list-number* | *list-name*] **delete** command entered in route-map configuration mode allows the deletion of a route target extended community attribute based on an extended community list.

Restrictions for MPLS route target rewrite

MPLS route target rewrite on switches is subjected to these restrictions:

- You can implement route target rewrite only in a single AS topology.
- Does not support **ip unnumbered** command in MPLS configuration.

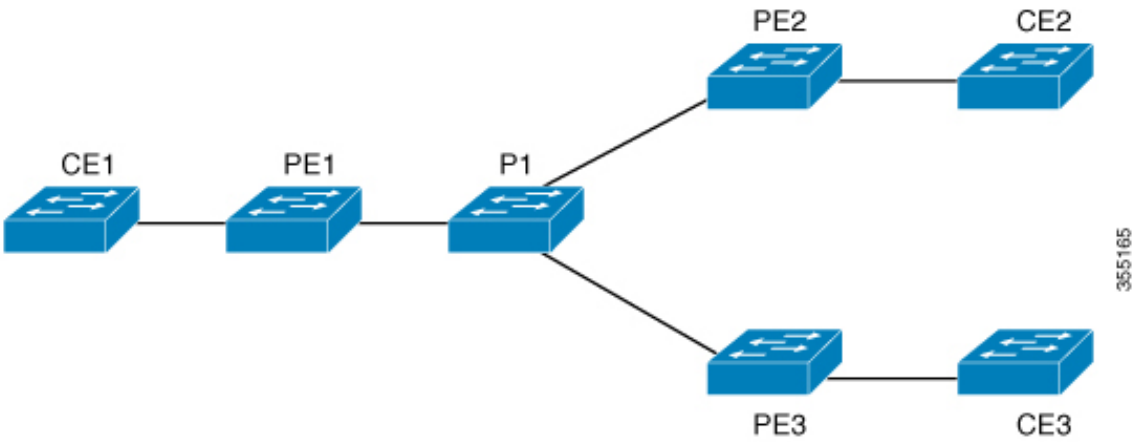
Configure route target replacement policy

Use this procedure to configure a route target replacement policy that involves creating an extended community list, defining a route map, and specifying the match and set actions for route target replacement.

You can configure the MPLS VPN route target rewrite feature on provider edge (PE) devices.

The figure shows an example of route target replacement on PE devices in an MPLS VPN single autonomous system topology.

Figure 1: Route target replacement on provider edge devices in a single MPLS VPN autonomous system topology



The example includes these 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.

Before you begin

To implement MPLS VPN route target rewrite, you must

- be familiar with configuring MPLS VPNs, and
- identify the RT replacement policy and target device for the autonomous system (AS).

If you configure a PE device to rewrite RT x to RT y and the PE has a VRF instance that imports RT x, you must configure the VRF to import RT y in addition to RT x.

Procedure

Step 1 Create an extended community access list and control access to it.

Example:

```
Device> enable
Device# configure terminal
Device(config)# ip extcommunity-list 1 permit rt 65000:2
```

Specify the respective input values for each of the command options as applicable:

- *standard-list-number* : identifies one or more permit or deny groups of extended communities
- *expanded-list-number*: identifies one or more permit or deny groups of extended communities. Regular expressions can be configured with expanded lists, but not standard lists.
- **permit**: permits access for a matching condition
- **deny**: denies access for a matching condition
- *regular-expression* : 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.
- **rt**: specifies the route target extended community attribute. The **rt** keyword can be configured only with standard extended community lists and not expanded community lists.
- **soo**: 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.
- *extended-community-value*: specifies the route target or site of origin. The value can be one of the these combinations:
  - *autonomous-system-number*:*network-number*
  - *ip-address*:*network-number*

  The colon is used to separate the combination values.

Step 2 Define the conditions for redistributing routes or enable policy routing, and enter route-map configuration mode.

Example:

```
Device(config)# route-map test-rtrewrite-map permit 10
```

Specify these values:

- Route map name: Route maps define conditions for redistributing routes between routing protocols or enabling policy routing. Multiple route maps can share the same map name. The **redistribute** router configuration command uses this name to reference the route map.
- Action: **permit** or **deny**
- Sequence number: that indicates the position a new route map will have in the list of route maps already configured with the same name.

This table describes the conditional logic and actions of route maps:

If...	And...	Then...
the match criteria are met for the route map	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.
the match criteria are not met for the route map	the **permit** keyword is pecified	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 match criteria are met for the route map	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.

Step 3 Match the BGP extended community list attributes by specifying the *standard-list-number* and *expanded-list-number* values.

Example:

```
Device(config-route-map)# match extcommunity 1
```

Example:

```
Device(config-route-map)# match extcommunity 101
```

The arguments identify one or more permit or deny groups of extended community attributes.

Step 4 Remove a route target from an extended community attribute of an inbound or outbound BGP VPNv4 update by specifying the extended community list number.

Example:

```
Device(config-route-map)# set extcomm-list 1 delete
```

The arguments identify one or more permit or deny groups of extended community attributes.

Step 5 Set BGP extended community attributes.

Example:

```
Device(config-route-map)# set extcommunity rt 65000:1 additive
Device(config-route-map)# end
```

The **additive** keyword adds a route target to the existing route target list without replacing any existing route targets.

Step 6 Verify that the match and set entries are correct.

Example:

```
Device# show route-map test-rtrewrite-map
```

A route target replacement policy is configured on the device.

What to do next

Associate route maps with specific BGP neighbors.

Apply route target replacement policy

Use this procedure to apply route target replacement policy by associating route maps with specific BGP neighbors.

Procedure

Step 1 Configure a BGP routing process by specifying the autonomous system number and place the device in router configuration mode.

Example:

```
Device> enable
Device# configure terminal
Device(config)# router bgp 100
```

The autonomous system number helps identify the device to other BGP devices and tags the routing information that is passed along.

Step 2 Add an entry to the BGP or multiprotocol BGP neighbor table by specifying the IP address of the neighbor, BGP peer group, and the autonomous system to which the neighbor belongs.

Example:

```
Device(config-router)# neighbor 172.10.0.2 remote-as 100
```

Step 3 Enter address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.

Example:

```
Device(config-router)# address-family vpnv4
```

Use the optional **unicast** keyword to specify VPNv4 unicast address prefixes.

Step 4 Enable the exchange of information with a neighboring BGP device by specifying the IP address of the neighbor and the BGP peer group name.

Example:

```
Device(config-router-af)# neighbor 172.16.0.2 activate
```

Step 5 Specify that a community attribute must be sent to a BGP neighbor.

Example:

```
Device(config-router-af)# neighbor 172.16.0.2 send-community extended
```

Use **standard**, **extended**, or **both** keyword respectively to send standard, extended, or both community attribute(s).

Step 6 Apply a route map to incoming or outgoing routes.

Example:

```
Device#
Device(config-router-af)# neighbor 172.16.0.2 route-map test-rtrewrite-map in
Device(config-router-af)# end
```

Use **in**, or **out** keyword respectively to apply route map to incoming or outoging routes.

The route target replacement policy is applied to the specified BGP neighbor.

Verify route target replacement

Use this procedure to verify that the VPNv4 prefixes with a specified route target extended community attribute are replaced correctly.

Procedure

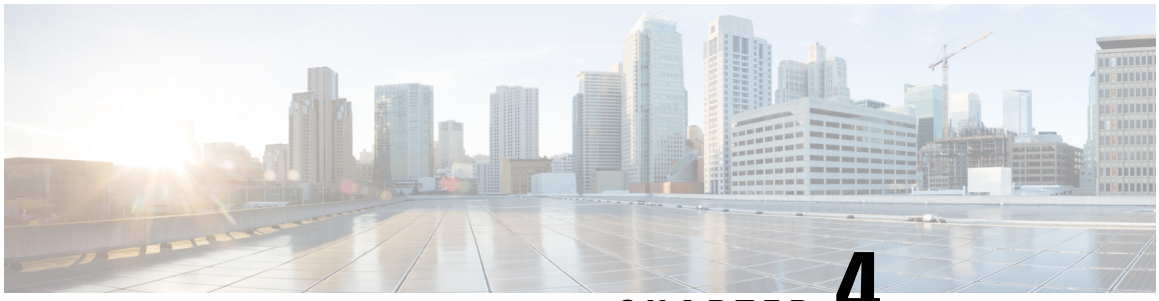
Verify the BGP VPNv4 routing table entry within a specific VRF on the provide edge device.

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
```

The output confirms whether the RT extended community attributes have been replaced as expected.



CHAPTER 4

MPLS Layer 3 VPN

An MPLS Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of an MPLS provider core network. At each customer site, one or more customer edge (CE) devices attach to one or more provider edge (PE) devices. This module explains the basic concepts of MPLS L3VPN and how to configure it on Cisco Smart Switches.

Feature history for MPLS Layer 3 VPN

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS Layer 3 VPN	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS virtual private networks

An MPLS virtual private network (VPN) is a Layer 3 IP-based network that

- delivers private network services over a public or shared infrastructure
- utilizes a peer model to exchange routing information between the service provider and the customer
- functions at Layer 3 and allows the service provider to relay data between customer sites without direct customer involvement in the core routing, and
- simplifies management by requiring updates only at the provider edge device when adding new sites.

A VPN is

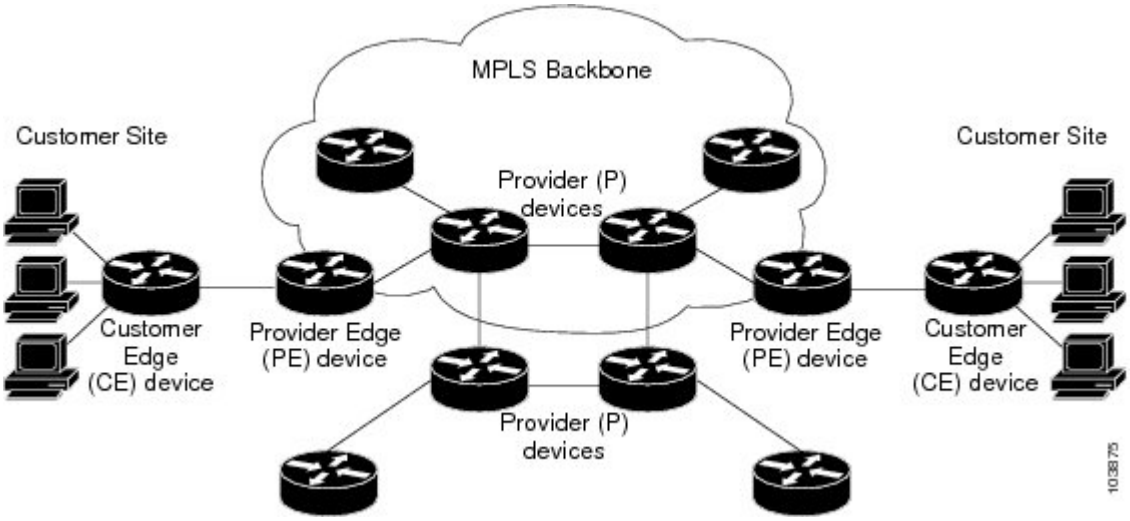
- an IP-based network delivering private network services over a public infrastructure, and
- a set of sites that communicate with each other privately over the Internet or other public or private networks.

Provider and customer devices are network components that

- define the architecture of an MPLS VPN
- assign or process MPLS and VPN labels, and
- establish the boundaries between the service provider core and the customer network.

Basic MPLS VPN terminology

Figure 2: Sample MPLS VPN topology



Different components of the MPLS VPN topology include:

- Provider device (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 sub-interface on which they are received. A PE device attaches directly to a customer edge (CE) device.
- Customer device (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.

MPLS VPNs and conventional VPNs

The table compares the characteristics of conventional VPNs and MPLS-based VPNs.

Table 1: MPLS VPNs Vs. conventional VPNs

Attribute	Conventional VPN	MPLS VPN
Creation method	Created by configuring a full mesh of tunnels or permanent virtual circuits (PVCs) to all sites	Created in Layer 3 and based on the peer model
Routing model	Requires a connection-oriented, point-to-point overlay on the network	Enables the service provider and customer to exchange Layer 3 routing information
Data relaying	Relies on pre-established tunnels or circuits managed by the customer	The service provider relays data between customer sites without the customer involvement
Management and expansion	Difficult to maintain or expand	Easier to manage and expand than conventional models
Adding new sites	Requires configuration changes on every edge device within the VPN	Requires an update only to the service provider's edge device that connects the new site

Benefits of MPLS virtual private network

MPLS virtual private networks allow service providers to deploy scalable VPNs. They build the foundation to deliver several value-added services.

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, and

- 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 PE device as opposed to all other 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 single 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 these areas:

- at the edge of a provider network, ensuring packets that are received from a customer are placed on the correct VPN, and
- 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 sub-interface 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, and
- 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.

Restrictions for MPLS virtual private networks

Follow the restrictions mentioned in this section when configuring static routes in an MPLS or MPLS VPN environment.

Generic restrictions for MPLS VPN

- Does not support MPLS fragmentation
- Does not support MPLS maximum transmission unit (MTU) configuration
- Supports only the default mode—per-VRF MPLS label allocation mode. However, they can inter-operate with remote peers operating in the per-prefix mode.

Supported static routes in an MPLS environment

When static routes are configured in an MPLS or MPLS VPN environment, only some variations of the **ip route** and **ip route vrf** commands are supported.

This variation of **ip route** command is supported when you configure static routes in an MPLS environment:

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

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

Unsupported static routes in an MPLS environment that uses the TFIB

Certain static routes are not supported in an MPLS environment that uses the TFIB.

- This variation of **ip route** command is not supported when you configure static routes in an MPLS environment:

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

- This variation of **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*

- These variations of **ip route** command 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**

Supported static Routes in an MPLS VPN environment

Only certain static routes are supported in an MPLS VPN environment.

- These variations of **ip route vrf** command 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**
- These variations of **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.)

Unsupported static routes in an MPLS VPN environment that uses the TFIB

Certain static routes are not supported in an MPLS environment that uses the TFIB.

- This variation of **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**

- These variations of **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**

- These variations of **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

Only certain static routes are supported in an MPLS VPN environment where the next hop resides in the global table on the CE device.

- This variation of **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) multi-hop cases.

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

- This variation of **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**

How MPLS virtual private networks work

MPLS VPN functionality is enabled at the edge of an MPLS network.

Summary

These are the key components involved in this process:

- Provider Edge (PE) device: translates customer routing information into VPNv4 routes and exchanges them with other PE devices
- Customer Edge (CE) device: exchanges standard routing updates with the PE device.
- VPN route target communities: lists used to identify all members of a VPN community. VPN route targets need to be configured for each VPN community member.
- Multiprotocol BGP (MP-BGP): propagates virtual routing and forwarding (VRF) reachability information

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.

Workflow

These stages describe how MPLS VPNs work:

1. Routing update exchange: The PE device exchanges routing updates with the connected CE device.
2. Route translation: The PE device translates the received CE routing information into VPNv4 routes.
3. VPNv4 propagation: The PE device uses MP-BGP to exchange these VPNv4 routes with other PE devices in the network.
4. Label attachment: The PE device attaches VPN labels .
5. MPLS forwarding: The provider core transports the traffic between VPN members using MPLS labels.

Result

Customer sites can communicate privately across a shared provider core as if they were on a single private network.

Configure an MPLS virtual private network

Before you begin

- Make sure that you have installed Multiprotocol Label Switching (MPLS), Label Distribution Protocol (LDP), and Cisco Express Forwarding (CEF) in your network.
- All devices in the core, including the provider edge (PE) devices, must be able to support CEF and MPLS forwarding. See the step to assess the needs of MPLS VPN customers.
- Enable CEF on all devices in the core, including the PE devices.
- 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.

Procedure

Step 1 Assess the needs of MPLS VPN customers.

Step 2 Enable MPLS on all devices in the core by configuring MPLS Label Distribution Protocol (LDP).

For configuration information, see the *MPLS Label Distribution Protocol* chapter.

Step 3 Connect the MPLS VPN customers.

a) Enable customer connectivity by defining VRFs on the provider edge (PE) devices.
b) Configure VRF interfaces on PE devices for each customer.
c) Configure routing protocols between the PE and customer edge (CE) devices.

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

Step 4 Verify the VPN configuration using the **show ip vrf** command.

A route distinguisher (RD) must be configured for the VRF instance.

The command output displays the set of defined VRF instances and associated interfaces. It also maps the VRF instances to the configured RD.

Step 5 Verify connectivity between MPLS VPN sites.

Assess core network requirements

This task identifies the core network topology to best serve MPLS VPN customers.

Perform this assessment before configuring a MPLS VPN to ensure that the infrastructure supports the required scale and features.

Procedure

Step 1 Identify the size of the network.

Identify these parameters to determine the number of devices and ports that you need:

- the number of customers you need to support
- the number of VPNs needed per customer, and
- the number of virtual routing and forwarding instances 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 (HA) support.

Step 4 Determine if you need Border Gateway Protocol (BGP) load sharing and redundant paths in the MPLS VPN core.

MPLS VPN Nonstop Forwarding and Graceful Restart are supported on selected devices and Cisco software releases. Contact Cisco Support for the exact requirements and hardware support.

What to do next

Configure MPLS in the core.

To enable MPLS on all devices in the core, you must configure MPLS Label Distribution Protocol (LDP).

Define VRFs on PE devices

This task defines a virtual routing and forwarding (VRF) configuration for IPv4.

Use this procedure on provider edge (PE) devices to segregate customer traffic and enable private connectivity.

Procedure

Step 1 Define the virtual private network (VPN) routing instance by assigning a VRF name.

Example:

```
Device> enable
Device# configure terminal
Device(config)# vrf definition vrf1
```

`vrf1` in this example is the unique name assigned to a VRF.

Step 2 Create routing and forwarding tables

Example:

```
Device(config-vrf)# rd 100:1
```

The route-distinguisher (RD) argument adds an 8-byte value to an IPv4 prefix to create a unique VPN IPv4 prefix.

You can enter a route distinguisher (RD) in formats such as 16-bit AS number:32-bit number such as, 101:3, or 32-bit IP address:16-bit number such as, 10.0.0.1:1.

Step 3 Create a route-target extended community for a VRF to control the distribution of routing information.

Example:

```
Device(config-vrf)# address-family ipv4
Device(config-vrf-af)# route-target both 100:1
Device(config-vrf-af))# exit
```

You can use the **import** keyword to import routing information from the target VPN community, **export** keyword to export it, or **both** to perform both the actions.

100:1 in this example is the *route-target-ext-community* argument that adds the route-target extended community attributes to the VRF's list of import, export, or both route-target extended communities.

Configure VRF interfaces on PE devices

This task associates a virtual routing and forwarding (VRF) instance with an interface or sub-interface.

Perform this task on provider edge (PE) devices for each VPN customer to link physical or logical interfaces to the correct VRF.

Procedure

Associate a VRF with an interface on the PE device.

Example:

```
Device> enable
```

```
Device# configure terminal
Device(config)# interface GigabitEthernet 0/0/1
Device(config-if)# vrf forwarding vrf1
Device(config-if)# end
```

What to do next

Verify the VPN configuration.

Verify connectivity between MPLS VPN sites

Procedure

Step 1 Verify IP connectivity from CE device to CE device across the MPLS core.

a) Diagnose basic network connectivity on various networks using the **ping** command along with the respective protocol, and host-name or system-address arguments.

Example:

```
Device> enable
Device# pingprotocol {host-name | system-address
```

b) Discover the routes that packets take when traveling to their destination using the **trace** command.

The **trace** command can help isolate a trouble spot if two devices cannot communicate.

c) Display the current state of the routing table using the **show ip route** command.

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.

Step 2 Verify that the local and remote CE devices are in the PE routing table.

a) Display the IP routing table that is associated with a VRF instance using the **show ip route vrf** command.

Check if the loopback addresses of the local and remote CE devices are in the routing table of the provider edge (PE) devices.

b) Display the Cisco Express Forwarding (CEF) forwarding table that is associated with a VRF using the **show ip cef vrf** command.

Check if the prefix of the remote CE device is in the CEF table.

Configuration examples for MPLS virtual private network

Configuration example for MPLS VPN using RIP

This section provides a sample configuration for MPLS VPN using RIP.

PE 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
mpls ldp graceful-restart
!
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
 no bgp default ipv4-unicast
!
address-family vpnv4
 neighbor 10.0.0.3 activate
 neighbor 10.0.0.3 send-community extended
 exit-address-family
!
address-family ipv4 vrf vpn1
 redistribute connected
```

```
 redistribute rip
 no auto-summary
 no synchronization
 exit-address-family
```

CE configuration

```
ip cef
mpls ldp router-id Loopback0 force
!
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
```

Configuration example for MPLS VPN using static routes

This section provides a sample configuration for MPLS VPN using static routes.

PE 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
mpls ldp graceful-restart
!
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
 no bgp default ipv4-unicast
 !
address-family vpnv4
 neighbor 10.0.0.3 activate
 neighbor 10.0.0.3 send-community extended
 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
```

CE configuration

```
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
```

Configuration example for MPLS VPN using BGP

This section provides a sample configuration for MPLS VPN using BGP.

PE configuration

```
router bgp 5001
bgp log-neighbor-changes
bgp graceful-restart
no bgp default ipv-unicast
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
!
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
  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
```

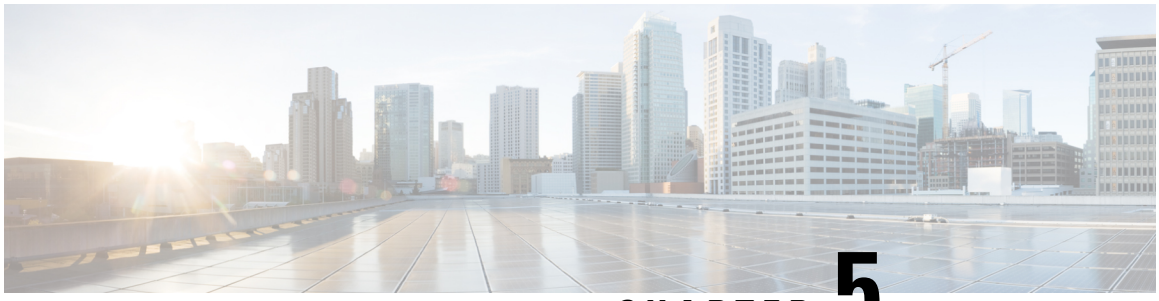
CE configuration

```
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
```

PART I

MPLS LDP



CHAPTER 5

MPLS Label Distribution Protocol

MPLS Label Distribution Protocol (LDP) enables peer label switch routers (LSRs) in an Multiprotocol Label Switching (MPLS) network to exchange label binding information for supporting hop-by-hop forwarding in an MPLS network. This module explains the concepts related to MPLS LDP and describes how to configure MPLS LDP in a network.

Feature history for MPLS Label Distribution Protocol

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS Label Distribution Protocol	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS Label Distribution Protocol

Multiprotocol Label Switching (MPLS) Label Distribution Protocol (LDP) is a signaling protocol that

- enables label switch routers (LSRs) in an MPLS network to discover potential peers and establish sessions with them to exchange label binding information
- provides the means for LSRs to request, distribute, and release label prefix binding information to peer devices, and
- supports hop-by-hop forwarding in an MPLS network.

Hop-by-hop forwarding: MPLS LDP allows one LSR to inform another LSR of the label bindings it has made. Once a pair of devices communicate the LDP parameters, they establish a label switched path (LSP). MPLS LDP distributes labels along normally routed paths to support MPLS forwarding, a method also known as hop-by-hop forwarding. MPLS LDP is useful for applications that require hop-by-hop forwarding, such as MPLS VPNs.

IP forwarding: When a packet arrives at a device, the device looks at the destination address in the IP header, performs a route lookup, and forwards the packet to the next hop.

MPLS forwarding: When a packet arrives at a device, the device looks at the incoming label, looks up the label in a table, and then forwards the packet to the next hop.

LDP sessions

LDP sessions are communication channels that

- label switch routers (LSRs) establish to exchange label binding information
- are initiated when LSRs send out messages to discover other LSRs, and
- can be directly connected or nondirectly connected.

Types of LDP sessions

There are two types of LDP sessions:

- Directly connected LDP sessions: that are established between LSRs that are one hop from each other, using link Hello messages to discover neighbors and negotiate session parameters for label exchange
- Indirectly connected LDP sessions: that are established between LSRs that are more than one hop apart, using targeted Hello messages for extended discovery. They enable label distribution for applications like MPLS Traffic Engineering (TE) tunnels.

Basic elements of an LDP session

This topic describes the basic elements of an LDP session—LDP label bindings, LDP label spaces, and LDP identifiers.

LDP label bindings

An LDP label binding is an association between a destination prefix and a label.

LDP label spaces

A set of possible labels from which a label used in a label binding is allocated from is called a label space.

LDP supports two types of label spaces:

- Interface-specific: An interface-specific label space uses interface resources for labels. For example, label-controlled ATM (LC-ATM) interfaces use virtual path identifiers (VPIs) or virtual circuit identifiers (VCIs) for labels. Depending on its configuration, an LDP platform may support zero, one, or more interface-specific label spaces.

- Platform-wide: An LDP platform supports a single platform-wide label space for use by interfaces that can share the same labels. For Cisco platforms, all interface types, except LC-ATM, use the platform-wide label space.

LDP identifier

LDP uses a 6-byte quantity called an LDP identifier (LDP ID) to name label spaces. The LDP ID is made up of these components:

- LPD router ID: the first four bytes that identify the label switch router (LSR) that owns the label space.
- Local label space ID: The last two bytes that identify the label space within the LSR. For the platform-wide label space, the last two bytes of the LDP ID are always both 0.

The LDP ID takes this form: *LDP router ID* : *local label space ID*. Examples: 172.16.0.0:0, 192.168.0.0:3

The **mpls ldp router-id** command allows you to specify the IP address of an interface as the LDP router ID.

The device determines the LDP router ID as follows, if the **mpls ldp router-id** command is not executed:

1. The device examines the IP addresses of all operational interfaces.
2. If these IP addresses include loopback interface addresses, the device selects the largest loopback address as the LDP router ID.
3. Otherwise, the device selects the largest IP address pertaining to an operational interface as the LDP router ID.

The normal or default method for determining the LDP router ID may result in a router ID that is not usable in certain situations. For example, the device might select an IP address as the LDP router ID that the routing protocol cannot advertise to a neighboring device.

Restriction for MPLS LDP sessions

You cannot enable MPLS LDP on a Virtual Routing and Forwarding (VRF) interface.

How directly connected LDP sessions work

Summary

The key components involved in the process are:

- Label Switch Router (LSR): sends and receives LDP Hello messages, establishes TCP connections, and negotiates session parameters
- LDP link Hello messages: User Datagram Protocol (UDP) packets sent through multicast to all devices on a subnet for basic discovery

Workflow

These stages describe how LSRs discover each other and establish LDP sessions for label exchange:

1. Discovery: An LSR sends LDP link Hello messages (UDP multicast) to all devices on its subnet.
2. Response: A neighboring LSR responds to the link Hello message, allowing the two devices to begin establishing an LDP session.
3. Role determination: Devices compare their transport addresses; the device with the higher IP address takes the active role, establishing the LDP TCP connection.
4. Session negotiation: After the TCP connection is established, LSRs negotiate session parameters, including the label distribution method.
5. Label distribution: LSRs exchange label binding information using either
   - Downstream unsolicited: where an LSR advertises label mappings to peers without being asked, or
   - Downstream on demand: where an LSR advertises label mappings to a peer only when the peer requests them.

Result

A directly connected LDP session is established, enabling the exchange of label binding information between neighboring LSRs.

How indirectly connected LDP sessions work

Summary

These are the key components and commands involved in establishing indirectly connected LDP sessions:

- Label Switch Router (LSR): sends and receives targeted LDP Hello messages, establishes TCP connections, and exchanges label binding information
- Targeted Hello messages: UDP unicast messages specifically addressed to a remote LSR for extended discovery
- **mpls ldp neighbor targeted** command: allows setting up targeted sessions when other means like, TE tunnels or AToM VCs do not apply, or to improve label convergence time for directly connected neighbors when links are unstable
- **mpls ldp discovery targeted-hello accept** command: configures an LSR to respond to requests for targeted Hello messages, allowing it to act as a passive participant

Workflow

These stages describe how indirectly connected LDP sessions are established and maintained:

1. Extended Discovery: An LSR sends a targeted Hello message, UDP unicast, specifically addressed to an indirectly connected neighbor.
2. Response: The indirectly connected LSR responds to the targeted Hello message, initiating the establishment of an LDP session.
3. Session establishment: A Multiprotocol Label Switching (MPLS) LDP targeted session, which is a label distribution session, is established between the devices.

4. Role assignment: The exchange of targeted Hello messages can involve different roles:
   - Active/Passive: Device 1 sends a targeted Hello with a response request; device 2 responds if configured to do so (passive).
   - Both Active: Both device 1 and device 2 send targeted Hello messages to each other.
5. Protocol mandate: The active LSR dictates the protocol for the targeted session, and the passive LSR adopts the protocol of the received targeted Hello messages
6. Convergence improvement (Optional): The **mpls ldp neighbor** *ip-address* **targeted** command can maintain sessions through alternate routes if direct links fail, allowing LSRs to retain and quickly reinstall labels.

Result

An indirectly connected LDP session is established, enabling label distribution between distant LSRs, which is crucial for applications like MPLS Traffic Engineering tunnels.

Configure MPLS LDP sessions

Establish directly connected MPLS LDP sessions

Use this procedure to establish MPLS LDP sessions between two directly connected devices.

This task involves enabling MPLS hop-by-hop forwarding globally and on specific interfaces, then configuring LDP as the label protocol for all interfaces.

Before you begin

Enable Cisco Express Forwarding on the device.

Procedure

Step 1 Configure MPLS hop-by-hop forwarding globally.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
```

The **mpls ip** command is enabled by default. You do not have to specify this command.

Globally enabling MPLS forwarding does not enable it on the device interfaces. You must enable MPLS forwarding on the interfaces as well as for the device.

Step 2 Configure LDP as the label protocol for all interfaces.

Example:

```
Device(config)# mpls label protocol ldp
```

The options available for this command depends on the hardware platform.

You can override this global setting for specific interfaces by specifying the command in interface configuration mode with either the **tdp** or **both** keyword.

Step 3 Configure MPLS hop-by-hop forwarding on a specific interface by entering the interface configuration mode.

Example:

```
Device(config)# interface HundredGigE 0/3/0
Device(config-if)# mpls ip
Device(config-if)# end
```

You must enable MPLS forwarding on the interfaces as well as for the device.

Step 4 Verify that directly connected MPLS LDP sessions are established, allowing label distribution between adjacent devices.

a) Verify that the interfaces have been configured to use LDP.

Example:

```
Device# show mpls interfaces
Interface                  IP            Tunnel   BGP Static Operational
HundredGigE0/3/0           Yes (ldp)     No       No  No     Yes
HundredGigE0/3/1           Yes           No       No  No     Yes
```

b) Verify that the interface is up and is sending Discovery Hello messages, as opposed to TDP Hello messages.

Example:

```
Device# show mpls ldp discovery
Local LDP Identifier:
    172.16.12.1:0
    Discovery Sources:
    Interfaces:
        HundredGigE0/3/0 (ldp): xmit
```

c) Display the status of LDP sessions.

Example:

```
Device# show mpls ldp neighbor
Peer LDP Ident: 10.1.1.2:0; Local LDP Ident 10.1.1.1:0
TCP connection: 10.1.1.2.18 - 10.1.1.1.66
State: Oper; Msgs sent/rcvd: 12/11; Downstream
Up time: 00:00:10
LDP discovery sources:
HundredGigE0/1/0, Src IP addr: 10.20.10.2
Addresses bound to peer LDP Ident:
10.1.1.2    10.20.20.1    10.20.10.2
```

Establish indirectly connected LDP sessions

Follow this procedure to establish MPLS LDP sessions between devices that are not directly connected.

This task involves enabling MPLS hop-by-hop forwarding globally and on specific interfaces, then configuring LDP as the label protocol for all interfaces.

This task explains how to configure indirectly connected MPLS LDP sessions, typically over a tunnel interface, to enable label distribution between distant Label Switch Routers (LSRs).

Before you begin

- Enable Cisco Express Forwarding (CEF) on the device.
- Configure the devices at both ends of the tunnel to be active or enable one device to be passive with the **mpls ldp discovery targeted-hello accept** command.

Procedure

Step 1 Configure MPLS hop-by-hop forwarding globally.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
```

The **mpls ip** command is enabled by default. You do not have to specify this command.

Globally enabling MPLS forwarding does not enable it on the device interfaces. You must enable MPLS forwarding on the interfaces as well as for the device.

Step 2 Configure LDP as the label protocol for all interfaces.

Example:

```
Device(config)# mpls label protocol ldp
```

The options available for this command depends on the hardware platform.

You can override this global setting for specific interfaces by specifying the command in interface configuration mode with either the **tdp** or **both** keyword.

Step 3 Configure a tunnel interface and assign an IP address to the interface.

Example:

```
Device(config)# interface tunnel 1
Device(config-if)# tunnel destination 172.16.1.1
```

Step 4 Configure MPLS hop-by-hop forwarding on the interface.

Example:

```
Device(config-if)# mpls ip
Device(config-if)# end
```

You must enable MPLS forwarding on the interfaces as well as for the device.

Step 5 Verify that indirectly connected MPLS LDP sessions are established, allowing label distribution between distinct LSRs.

Example:

```
Device# show mpls ldp discovery
Local LDP Identifier:
          172.16.0.0:0
Discovery Sources:
```

```
Interfaces:
POS1/2/0 (ldp): xmit/recv
LDP Id: 172.31.255.255:0
Tunnel1 (ldp): Targeted -> 192.168.255.255
Targeted Hellos:
172.16.0.0 -> 192.168.255.255 (ldp): active, xmit/recv
LDP Id: 192.168.255.255:0
172.16.0.0 -> 192.168.0.0 (ldp): passive, xmit/recv
LDP Id: 192.168.0.0:0
```

The command output indicates these:

- The LSR 172.16.0.0 sent LDP link Hello messages on interface POS1/2/0 and discovered neighbor 172.31.255.255.
- The local LSR sent LDP targeted Hello messages associated with interface Tunnel1 to target 192.168.255.255. The LSR was configured to use LDP.
- The local LSR is active for targeted discovery activity with 192.168.255.255; this means that the targeted Hello messages it sends to 192.168.255.255 carry a response request. The local LSR was configured to have an LDP session with the nondirectly connected LSR 192.168.255.255.
- The local LSR is not passive from the discovery activity with 192.168.255.255 for one of these reasons:
  - The targeted Hello messages it receives from 192.168.255.255 do not carry a response request.
  - The local LSR has not been configured to respond to such requests.
- The local LSR sent Tag Distribution Protocol (TDP) directed Hello messages to the target LSR 192.168.0.0. This LSR uses TDP because the Hello messages received from the target LSR 192.168.0.0 were TDP directed Hello messages.
- The local LSR is passive in discovery activity with LSR 192.168.0.0. This means that the directed Hello messages it receives from LSR 192.168.0.0 carry a response request and that the local LSR has been configured with the **mpls ldp discovery targeted-hello accept** command to respond to such requests from LSR 192.168.0.0.
- The local LSR is not active in discovery activity with LSR 192.168.0.0, because no application that requires an LDP session with LSR 192.168.0.0 has been configured on the local LSR.

Preserve QoS settings of MPLS LDP packets

Use this procedure to preserve QoS settings of MPLS LDP packets in an LDP session.

Normally, LDP advertises an Implicit NULL label for directly connected routes, causing the penultimate LSR to remove the MPLS header. This removal can lead to the loss of QoS values before the packet reaches the last LSR. By advertising an Explicit NULL label, the LSR at the penultimate hop forwards MPLS packets with a NULL label (value of zero) instead of forwarding them as IP packets, thereby preserving the QoS information.

When you issue the **mpls ldp explicit-null** command, Explicit NULL is advertised in place of Implicit NULL for directly connected prefixes.

Note An explicit NULL label is not needed when the penultimate hop receives MPLS packets with a label stack that contains at least two labels and penultimate hop popping is performed. In that case, the inner label can still carry the QoS value needed by the penultimate and edge LSR to implement their QoS policy.

Procedure

Step 1 Establish directly connected MPLS LDP session on the switch by configuring MPLS hop-by-hop forwarding globally and at the interface-level. Also, configure the use of LDP on all interfaces.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
Device(config)# mpls label protocol ldp
Device(config-mpls)# exit
Device(config)# interface HundredGigE 0/3/0
Device(config-if)# mpls ip
Device(config-if)# end
```

Step 2 Configure the switch to advertise an Explicit Null label in situations where it would normally advertise an Implicit Null label.

- Use **mpls ldp explicit-null** command without any keywords.

  ```
  Device(config)# mpls ldp explicit-null
  ```

  Enabling Explicit NULL on an egress LSR causes that LSR to advertise the Explicit NULL label to all adjacent MPLS devices.

- Use **mpls ldp explicit-null** command with **for** keyword.

  ```
  Device(config)# mpls label protocol ldp
  Device(config)# access-list 24 permit host 10.24.24.24
  Device(config)# mpls ldp explicit-null for 24
  ```

  Enabling Explicit NULL with the **for** keyword with a standard access control list (ACL) changes all tables in the adjacent MPLS devices to swap an Explicit NULL label for only those entries specified in the access-list. In this example, an access-list is created that contains the 10.24.24.24/32 entry. Explicit NULL is configured for that access list.

- Use **mpls ldp explicit-null** command with **to** keyword.

  ```
  Device(config)# mpls label protocol ldp
  Device(config)# access-list 15 permit host 10.15.15.15
  Device(config)# mpls ldp explicit-null to 15
  ```

  Enabling Explicit NULL with the **to** keyword and an access list enables you to advertise Explicit NULL labels to only those adjacent devices specified in the access-list. To advertise Explicit NULL to a particular device, you must specify the LDP ID of the device in the access-list.

In this example, an access-list contains the 10.15.15.15/32 entry, which is the LDP ID of an adjacent MPLS device. The device that is configured with Explicit NULL advertises Explicit NULL labels only to that adjacent device.

- Use **mpls ldp explicit-null** command with both **for** and **to** keywords.

```
Device(config)# mpls ldp explicit-null for 24 to 15
```

Enabling Explicit NULL with both the **for** and **to** keywords allows you to specify which routes to advertise with Explicit NULL labels and to which adjacent devices to advertise these Explicit NULL labels.

Step 3 Verify that MPLS packets are forwarded with an Explicit NULL label, that is, value of zero.

Example:

Scenario 1: Explicit NULL configuration without any keyword

```
Device# show mpls forwarding-table

Local  Outgoing    Prefix            Bytes label Outgoing           Next Hop
label  label or VC or Tunnel Id      switched   interface
19     Pop tag     10.12.12.12/32    0          HundredGigE2/1/0    172.16.0.1
22     0           10.14.14.14/32    0          HundredGigE2/0/0    192.168.0.2
23     0           172.24.24.24/32   0          HundredGigE2/0/0    192.168.0.2
24     0           192.168.0.0/8     0          HundredGigE2/0/0    192.168.0.2
25     0           10.15.15.15/32    0          HundredGigE2/0/0    192.168.0.2
26     0           172.16.0.0/8      0          HundredGigE2/0/0    192.168.0.2
27     25          10.16.16.16/32    0          HundredGigE2/0/0    192.168.0.22
28     0           10.34.34.34/32    0          HundredGigE2/0/0    192.168.0.2
```

If you issue the **show mpls forwarding-table** command on an adjacent device, the output shows that the only the outgoing labels for the addresses specified (172.24.24.24/32) change from Pop label to 0. All other Pop label outgoing labels remain the same.

Scenario 2: Explicit NULL configuration with **for** keyword

```
Device# show mpls forwarding-table

Local  Outgoing    Prefix            Bytes label Outgoing           Next Hop
label  label or VC or Tunnel Id      switched   interface
19     Pop tag     10.12.12.12/32    0          HundredGigE2/1/0    172.16.0.1
22     0           10.14.14.14/32    0          HundredGigE2/0/0    192.168.0.2
23     0           172.24.24.24/32   0          HundredGigE2/0/0    192.168.0.2
24     0           192.168.0.0/8     0          HundredGigE2/0/0    192.168.0.2
25     0           10.15.15.15/32    0          HundredGigE2/0/0    192.168.0.2
26     0           172.16.0.0/8      0          HundredGigE2/0/0    192.168.0.2
27     25          10.16.16.16/32    0          HundredGigE2/0/0    192.168.0.22
28     0           10.34.34.34/32    0          HundredGigE2/0/0    192.168.0.2
```

Scenario 3: Explicit NULL configuration with **to** keyword

```
Device# show mpls forwarding-table

Local  Outgoing    Prefix            Bytes label Outgoing          Next Hop
label  label or VC or Tunnel Id      switched   interface
19     Pop tag     10.12.12.12/32    0          HundredGigE2/1/0    172.16.0.1
22     0           10.14.14.14/32    0          HundredGigE2/0/0    192.168.0.2
23     0           172.24.24.24/32   0          HundredGigE2/0/0    192.168.0.2
24     0           192.168.0.0/8     0          HundredGigE2/0/0    192.168.0.2
25     0           10.15.15.15/32    0          HundredGigE2/0/0    192.168.0.2
26     0           172.16.0.0/8      0          HundredGigE2/0/0    192.168.0.2
```

```
27     25          10.16.16.16/32    0          HundredGigE2/0/0    192.168.0.22
28     0           10.34.34.34/32    0          HundredGigE2/0/0    192.168.0.2
```

The output shows that Explicit NULL labels are going only to the device specified in the access list.

Scenario 4: Explicit NULL configuration with both **for** and **to** keywords

```
Device# show mpls forwarding-table

Local  Outgoing    Prefix            Bytes label Outgoing           Next Hop
label  label or VC or Tunnel Id      switched   interface
17     0  <---     10.24.24.24/32     0          HundredGigE2/0/0   172.16.0.1
20     Pop tag     172.16.0.0/8       0          HundredGigE2/0/0   172.16.0.1
21     20          10.12.12.12/32     0          HundredGigE2/0/0   172.16.0.1
22     16          10.0.0.0/8         0          HundredGigE2/0/0   172.16.0.1
23     21          10.13.13.13/32     0          HundredGigE2/0/0   172.16.0.1
25     Pop tag     10.14.14.14/32     0          HundredGigE2/0/0   172.16.0.1
27     Pop tag     192.168.0.0/8      0          HundredGigE2/0/0   172.16.0.1
28     25          10.16.16.16/32     0          HundredGigE2/0/0   172.16.0.1
29     Pop tag     192.168.34.34/32   0          HundredGigE2/0/0   172.16.0.1
```

The output shows that it receives explicit null labels for 10.24.24.24/32.

Configure MD5 authentication for LDP peers

You can enable authentication between two LDP peers, which verifies each segment sent on the TCP connection between the peers. Authentication uses the Message Digest 5 (MD5) algorithm to verify the integrity of the communication and authenticate the origin of the message.

To enable authentication, issue the **mpls ldp neighbor** *ip-address* **password** command. This causes the device to generate an MD5 digest for every segment sent on the TCP connection and check the MD5 digest for every segment received from the TCP connection.

Follow these guidelines for configuring MD5 authentication for LDP peers:

- You must configure authentication on both LDP peers using the same password; otherwise, the peer session is not established.
- When you configure a password for an LDP neighbor, the device tears down existing LDP sessions and establishes new sessions with the neighbor.
- If a device has a password configured for a neighbor, but the neighboring device does not, the LDP session will not establish. The system displays a `%TCP-6-BADAUTH: No MD5 digest...` message.
- If the two devices have different passwords configured, the LDP session will not establish. The system displays a `%TCP-6-BADAUTH: Invalid MD5 digest...` message.

Procedure

Step 1 Establish directly connected MPLS LDP session on the switch by configuring MPLS hop-by-hop forwarding globally and configuring the use of LDP on all interfaces.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
Device(config)# mpls label protocol ldp
```

Step 2 Specify authentication between two LDP peers.

Example:

```
Device(config)# mpls ldp neighbor 10.1.1.1 password test-password
Device(config)# end
```

Step 3 Display the status of LDP session and verify that MD5 authentication is used for the session.

Example:

```
Device# show mpls ldp neighbor

Peer LDP Ident: 10.1.1.2:0; Local LDP Ident 10.1.1.1:0
        TCP connection: 10.1.1.2.11118 - 10.1.1.1.646
        State: Oper; Msgs sent/rcvd: 12/11; Downstream
        Up time: 00:00:10
        LDP discovery sources:
          HundredGigE1/0/0, Src IP addr: 10.20.10.2
        Addresses bound to peer LDP Ident:
          10.1.1.2    10.20.20.1    10.20.10.2


Device# show mpls ldp neighbor 10.0.0.21 detail

Peer LDP Ident: 10.0.0.21:0; Local LDP Ident 10.0.0.22:0
        TCP connection: 10.0.0.21.646 - 10.0.0.22.14709; MD5 on
        State: Oper; Msgs sent/rcvd: 1020/1019; Downstream; Last TIB rev sent 2034
        Up time: 00:00:39; UID: 3; Peer Id 1;
        LDP discovery sources:
          HundredGigE1/1/0; Src IP addr: 172.16.1.1
            holdtime: 15000 ms, hello interval: 5000 ms
        Addresses bound to peer LDP Ident:
          10.0.0.21       10.0.38.28       10.88.88.2      172.16.0.1
          172.16.1.1
        Peer holdtime: 180000 ms; KA interval: 60000 ms; Peer state: estab
```

The output shows that MD5 is used for the LDP session.

Specify the LDP router ID

Use this procedure to set the IP address of a specific interface as the LDP router ID.

This task allows you to explicitly define the LDP router ID, overriding the default selection process, which can prevent issues with unusable router IDs.

Before you begin

- The specified interface must be operational before assigning it as the LDP router ID.
- If using a loopback interface, configure the IP address with a `/32` network mask and ensure that the routing protocol advertises the corresponding `/32` network.

Procedure

Step 1 Configure MPLS hop-by-hop forwarding globally.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
```

Globally enabling MPLS forwarding does not enable it on the device interfaces. You must enable MPLS forwarding on the interfaces as well as for the device.

Step 2 Configure LDP as the label protocol for all interfaces.

Example:

```
Device(config)# mpls label protocol ldp
```

You can override this global setting for specific interfaces by specifying the command in interface configuration mode with either the **tdp**, or **both** keyword.

Step 3 Specify the preferred interface to be set as the LDP router ID.

Example:

```
Device(config)# mpls ldp router-id HundredGigE 2/0/0
Device(config)# end
```

If you use the **force** keyword with the **mpls ldp router-id** command, the router ID takes effect more quickly. Although, the effect of the command depends on the current state of the specified interface, the forced change in the LDP router ID tears down any existing LDP sessions, releases label bindings learned through the LDP sessions, and interrupts MPLS forwarding activity associated with the bindings.

The table describes the actions when the **mpls ldp router-id** command is configured with and without the **force** keyword.

If you issue the mpls ldp router-id command...	And...	Then...
without the **force** keyword	—	the device select selects the IP address of the specified interface the next time it is necessary to select an LDP router ID, provided that the interface is operational. This happens typically the next time the interface is shut down or the address is configured.
with the **force** keyword	the interface is up or operational and if its IP address is not currently the LDP router ID	the device forcibly changes the LDP router ID to the IP address of the interface.

If you issue the mpls ldp router-id command...	And...	Then...
with the **force** keyword	the interface is down or not operational	the device forcibly changes the LDP router ID to the IP address of the interface when the interface transitions to up.

Step 4 Display and verify the LDP identifier for the local device.

Example:

```
Device# show mpls ldp discovery
Local LDP Identifier:
    10.15.15.15:0
Discovery Sources:
    Interfaces:
        HundredGigE0/3/0 (ldp): xmit/recv
            LDP Id: 10.14.14.14:0
```

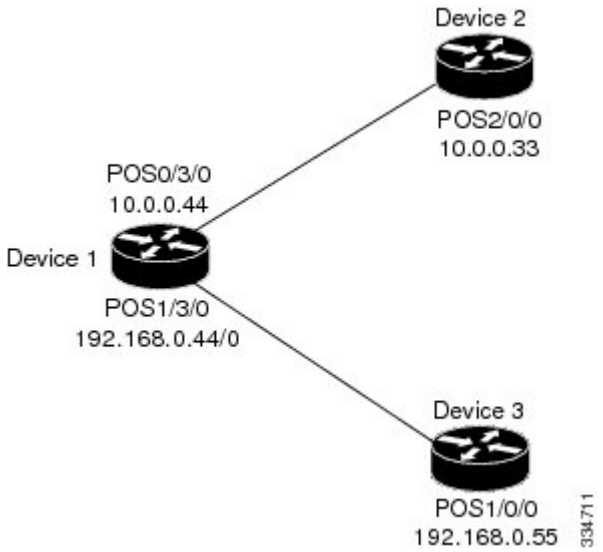
The LDP router ID is set to the IP address of the specified interface, ensuring a predictable and stable identifier for LDP operations.

Configuration Examples for MPLS LDP sessions

Configuration example for directly connected LDP sessions

This section provides a sample network and configurations for establishing directly connected LDP sessions.

Figure 3: Sample network topology for directly connected LDP sessions



In this example, we configure:

- MPLS hop-by-hop forwarding for the POS links between Device 1 and Device 2 and between Device 1 and Device 3
- LDP for label distribution between Device 1 and Device 2
- LDP for label distribution between Device 1 and Device 3
- A loopback interface and IP address for each LSR that can be used as the LDP router ID

Configuring the **mpls ip** command on an interface triggers the transmission of discovery Hello messages for the interface.

The LDP configuration for the devices uses the **mpls label protocol ldp** command in interface configuration mode. To specify LDP for all interfaces, use the **mpls label protocol ldp** command in global configuration mode without any **mpls label protocol** commands for the interface.

Configuration example: Device 1

```
ip cef distributed                                  !Assumes R1 supports distributed CEF
interface Loopback0                                 !Loopback interface for LDP ID
 ip address 172.16.0.11 255.255.255.255
!
interface HundredGigE0/3/0
 ip address 10.0.0.44 255.0.0.0
 mpls ip                                             !Enable hop-by-hop MPLS forwarding
 mpls label protocol ldp
!
interface HundredGigE1/3/0
 ip address 192.168.0.44 255.0.0.0
 mpls ip                                             !Enable hop-by-hop MPLS forwarding
 mpls label protocol ldp
```

Configuration example: Device 2

```
ip cef distributed                                  !Assumes R2 supports distributed CEF
!
interface Loopback0                                 !Loopback interface for LDP ID.
 ip address 172.16.0.22 255.255.255.255
 !
interface HundredGigE2/0/0
 ip address 10.0.0.33 255.0.0.0
 mpls ip                                             !Enable hop-by-hop MPLS forwarding
 mpls label protocol ldp
```

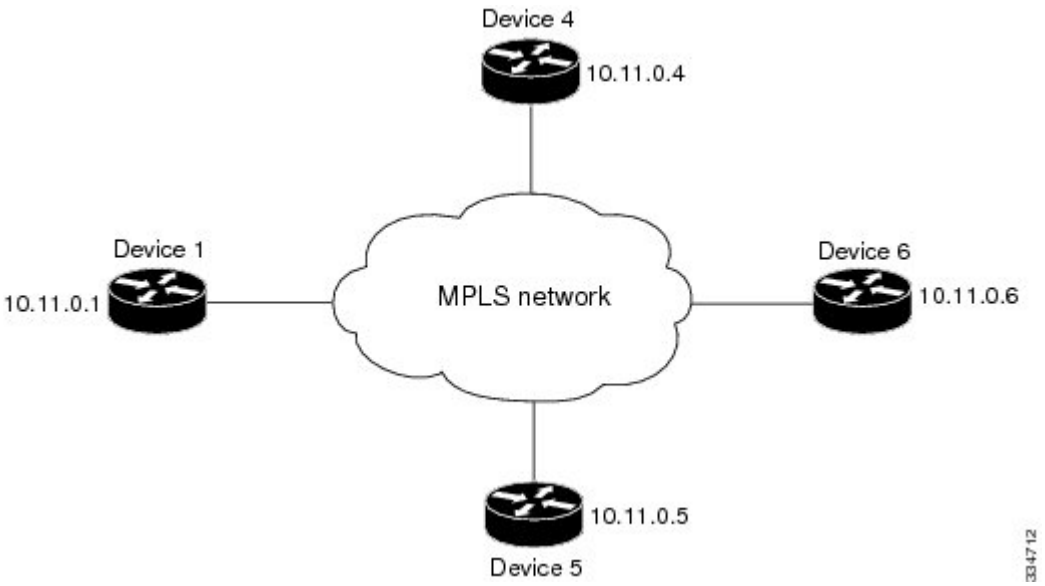
Configuration example: Device 3

```
ip cef                                              !Assumes R3 does not support dCEF
!
interface Loopback0                                 !Loopback interface for LDP ID.
 ip address 172.16.0.33 255.255.255.255
 !
interface HundredGigE1/0/0
 ip address 192.168.0.55 255.0.0.0
 mpls ip                                             !Enable hop-by-hop MPLS forwarding
 mpls label protocol ldp
```

Configuration example for indirectly connected LDP sessions

This section provides a sample network and configurations for establishing indirectly connected Label Distribution Protocol (LDP) sessions.

Figure 4: Sample network topology for indirectly connected LDP sessions



In this example:

- Targeted sessions between Devices 1 and 4 use LDP. Devices 1 and 4 are both active.
- Targeted sessions between Devices 1 and 6 use LDP. Device 1 is active and Device 6 is passive.
- Targeted sessions between Devices 1 and 5 use LDP. Device 5 is active.

These examples assume that the active ends of the indirectly connected sessions are associated with tunnel interfaces, such as MPLS traffic engineering tunnels. They show only the commands related to configuring LDP targeted sessions. The examples do not show configuration of the applications that initiate the targeted sessions.

Configuration example: Device 1

Tunnel interfaces Tunnel14 and Tunnel16 specify LDP for targeted sessions associated with these interfaces. The targeted session for Device 5 requires LDP. The **mpls label protocol ldp** command in global configuration mode makes it unnecessary to explicitly specify LDP as part of the configuration from the Tunnel14 and Tunnel16.

```
ip cef distributed             !Device1 supports distributed CEF
mpls label protocol ldp        !Use LDP for all interfaces

interface Loopback0            !Loopback interface for LDP ID.
 ip address 10.25.0.11 255.255.255.255

interface Tunnel14             !Tunnel to Device 4 requiring label distribution
 tunnel destination 10.11.0.4  !Tunnel endpoint is Device 4
 mpls ip                       !Enable hop-by-hop forwarding on the interface
```

```
interface Tunnel15              !Tunnel to Device 5 requiring label distribution
 tunnel destination 10.11.0.5  !Tunnel endpoint is Device 5
 mpls label protocol ldp       !Use LDP for session with Device 5
 mpls ip                       !Enable hop-by-hop forwarding on the interface

interface Tunnel16              !Tunnel to Device 6 requiring label distribution
 tunnel destination 10.11.0.6  !Tunnel endpoint is Device 6
 mpls ip                       !Enable hop-by-hop forwarding on the interface
```

Configuration example: Device 4

The **mpls label protocol ldp** command in global configuration mode makes it unnecessary to explicitly specify LDP as part of the configuration for the Tunnel41 targeted session with Device 1.

```
ip cef distributed               !Device 4 supports distributed CEF
mpls label protocol ldp          !Use LDP for all interfaces

interface Loopback0              !Loopback interface for LDP ID.
 ip address 10.25.0.44 255.255.255.255

interface Tunnel41               !Tunnel to Device 1 requiring label distribution
 tunnel destination 10.11.0.1    !Tunnel endpoint is Device 1
 mpls ip                         !Enable hop-by-hop forwarding on the interface
```

Configuration example: Device 5

Device 5 uses LDP for all targeted sessions. Therefore, its configuration includes the **mpls label protocol ldp** command.

```
ip cef                        !Device 5 supports CEF
mpls label protocol ldp       !Use LDP for all interfaces

interface Loopback0           !Loopback interface for LDP ID.
 ip address 10.25.0.55 255.255.255.255

interface Tunnel51            !Tunnel to Device 1 requiring label distribution
 tunnel destination 10.11.0.1 !Tunnel endpoint is Device 1
 mpls ip                      !Enable hop-by-hop forwarding on the interface
```

Configuration example: Device 6

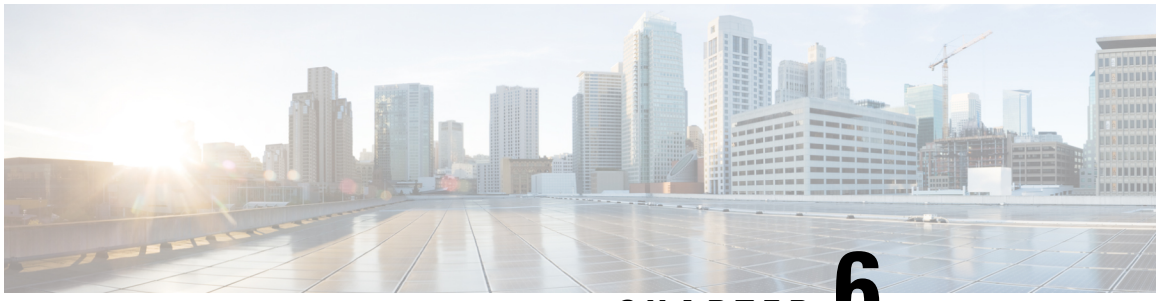
By default, a device cannot be a passive neighbor in targeted sessions. Therefore, Device 1, Device 4, and Device 5 are active neighbors in any targeted sessions. The **mpls ldp discovery targeted-hello accept** command permits Device 6 to be a passive target in targeted sessions with Device 1. Device 6 can also be an active neighbor in targeted sessions, although the example does not include such a configuration.

```
ip cef distributed                        !Device 6 supports distributed CEF

interface Loopback0                       !Loopback interface for LDP ID
 ip address 10.25.0.66 255.255.255.255

mpls ldp discovery targeted-hellos accept from LDP_SOURCES
                                          !Respond to requests for targeted hellos

                                          !from sources permitted by acl LDP_SOURCES
ip access-list standard LDP_SOURCES       !Define acl for targeted hello sources
 permit 10.11.0.1                         !Accept targeted hello request from Device 1
 deny any                                 !Deny requests from other sources
```



CHAPTER 6

MPLS LDP Session Protection

The MPLS LDP session protection feature provides faster Label Distribution Protocol (LDP) convergence when a link recovers following an outage. MPLS LDP session protection protects an LDP session between directly connected neighbors or an LDP session established for a traffic engineering (TE) tunnel. This module explains the concepts related to MPLS LDP session protection and describes how to configure MPLS LDP session protection in a network.

Feature history for MPLS LDP session protection

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS LDP session protection	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS LDP session protection

MPLS LDP session protection is an MPLS LDP feature that

- provides faster LDP convergence when a link recovers from an outage
- protects an LDP session between directly connected neighbors, and
- protects an LDP session established for a traffic engineering (TE) tunnel.

MPLS LDP session protection customization

This section explains how to customize MPLS LDP session protection feature.

You can modify MPLS LDP session protection by using specific keywords in the **mpls ldp session protection** command.

How long should an LDP Targeted Hello Adjacency be retained

The default behavior of the **mpls ldp session protection** command allows an LDP Targeted Hello Adjacency to exist indefinitely following the loss of an LDP Link Hello Adjacency. You can issue the **duration** keyword to specify the number of seconds that the LDP Targeted Hello Adjacency is retained after the loss of the LDP Link Hello Adjacency. When the link is lost, a timer starts. If the timer expires, the LDP Targeted Hello Adjacency is removed.

Which devices should have MPLS LDP session protection

The default behavior of the **mpls ldp session protection** command allows MPLS LDP session protection for all neighbor sessions. You can issue the **for** keyword to limit the number of neighbor sessions that are protected. You can create an access list that includes several peer devices. Specify that access list with the **for** keyword to enable LDP session protection for the peer devices in the access control list.

How MPLS LDP session protection works

MPLS LDP session protection feature allows LDP sessions to remain active during temporary link failures by leveraging Targeted Hello adjacency in addition to standard Link Hello adjacency.

This process is used in MPLS networks to improve LDP session resiliency and convergence time after link failures.

Summary

These are the key components involved in the MPLS LDP session protection process:

- Label switch routers (LSRs): network devices that send and receive LDP messages
- LDP Hello messages: used by LSRs to discover neighbors and establish LDP sessions
- LDP Link Hello: a UDP packet sent to all devices on a subnet for directly connected neighbors
- LDP Targeted Hello: a unicast UDP packet specifically addressed to a non-directly connected LSR
- LDP Link Adjacency: an LDP session between directly connected devices
- LDP Targeted Hello Adjacency: an LDP session established using Targeted Hellos, also used for session protection

MPLS LDP session protection uses LDP Targeted Hellos to protect LDP sessions. For example, two directly connected devices have LDP enabled and can reach each other through alternate IP routes in the network.

Workflow

These stages describe how MPLS LDP session protection works:

1. LSRs send LDP Hello messages to discover other LSRs for LDP session creation.

If...	Then...
LSRs are directly connected or one hop	they send LDP Link Hellos as UDP packets to all devices on the subnet. A neighboring LSR responds, and an LDP session is established, forming an LDP Link adjacency.
LSRs are not directly connected or more than one hop	they send LDP Targeted Hellos as unicast UDP packets to specific LSRs. The non-directly connected LSR responds, and an LDP session is established—a targeted session, often for traffic-engineered paths.

2. When MPLS LDP session protection is enabled, an LDP Targeted Hello adjacency is established for the LDP session in addition to the LDP Link Hello adjacency.

If...	Then...
the directly connected link between two devices fails	the LDP Link adjacency also fails.
if the LDP peer is still reachable through alternate IP routes in the network	the LDP session remains active because the LDP Targeted Hello adjacency persists between the devices.

3. When the directly connected link recovers, the LDP session does not need to be reestablished. And, LDP bindings for prefixes do not need to be relearned, leading to faster convergence.

Result

MPLS LDP session protection feature ensures LDP sessions remain active during temporary link failures, preventing the need for re-establishment and relearning of LDP bindings when the link recovers.

Restrictions for MPLS LDP session protection

The MPLS LDP session protection feature is not supported in certain scenarios:

- with extended access lists
- with LC-ATM devices, and
- with Tag Distribution Protocol (TDP) sessions.

Configure MPLS LDP session protection

This task involves configuring basic IP and MPLS settings before enabling LDP session protection to ensure faster LDP convergence when a link recovers following an outage.

Before you begin

Follow these guidelines for configuring MPLS LDP session protection:

- Label switch routers (LSRs) must be able to respond to LDP Targeted Hellos. Otherwise, the LSRs cannot establish a Targeted adjacency. All devices that participate in MPLS LDP session protection must be enabled to respond to Targeted Hellos.
- Both neighbor devices must be configured for session protection. If not, one device must be configured for session protection and the other device must be configured to respond to Targeted Hellos.

Procedure

Step 1 Configure Cisco Express Forwarding (CEF).

Example:

```
Device> enable
Device# configure terminal
Device(config)# ip cef
```

You can also configure distributed CEF.

Step 2 Configure a loopback interface and assign an IP address to the loopback interface.

Example:

```
Device(config)# interface Loopback 0
Device(config-if)# ip address 10.25.0.11 255.255.255.255
Device(config-if)# exit
```

Step 3 Configure MPLS hop-by-hop forwarding on an interface.

Example:

```
Device(config)# interface HundredGigE 1/0/1
Device(config-if)# mpls ip
Device(config-if)# mpls label protocol ldp
Device(config-if)# exit
Device(config)# exit
```

Step 4 Enable MPLS LDP session protection.

Example:

```
Device(config)# mpls ldp session protection
Device(config)# end
```

Entering the **mpls ldp session protection** command without a keyword protects all LDP sessions.

Specify the keywords for the command, as applicable:

- The **for acl** keyword and argument specifies a standard IP access control list (ACL) of prefixes to be protected.
- The **duration** keyword specifies how long the device should retain the LDP Targeted Hello Adjacency following the loss of the LDP Link Hello Adjacency.

- The **infinite** keyword specifies that the LDP Targeted Hello Adjacency should be retained forever after a link is lost.
- The **seconds** argument specifies the time in seconds that the LDP Targeted Hello Adjacency should be retained after a link is lost. The range is 30 to 2,147,483 seconds.

Verify MPLS LDP session protection

Use this procedure to confirm the operational status of MPLS LDP session protection feature.

Procedure

Step 1 Verify information of the LDP neighbors.

Example:

```
Device# show mpls ldp neighbor

Peer LDP Ident: 10.0.0.3:0; Local LDP Ident 10.0.0.5:0
TCP connection: 10.0.0.3.646 - 10.0.0.5.11005
State: Oper; Msgs sent/rcvd: 1453/1464; Downstream
Up time: 21:09:56
LDP discovery sources:
 Targeted Hello 10.0.0.5 -> 10.0.0.3, active
Addresses bound to peer LDP Ident:
 10.3.104.3       10.0.0.2        10.0.0.3
```

Check if the output contains the term `xmit/recv` for the peer device.

Step 2 Verify that the MPLS LDP session protection state is `Ready` or `Protecting`.

Example:

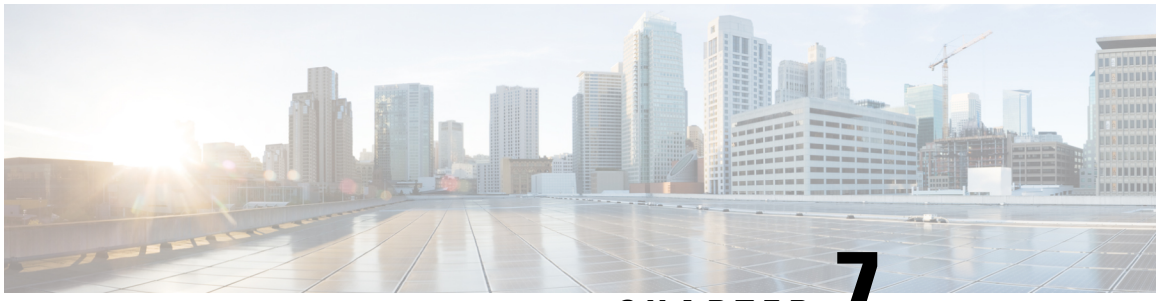
```
Device#show mpls ldp neighbor detail
    Peer LDP Ident: 44.44.44.44:0; Local LDP Ident 1.1.1.1:0
        TCP connection: 44.44.44.44.29723 - 1.1.1.1.646
        Password: not required, none, in use
        State: Oper; Msgs sent/rcvd: 11085/11089; Downstream; Last TIB rev sent 81942
        Up time: 00:17:46; UID: 3; Peer Id 2
        LDP discovery sources:
         Port-channel44; Src IP addr: 104.1.1.5
           holdtime: 15000 ms, hello interval: 5000 ms
         Targeted Hello 1.1.1.1 -> 44.44.44.44, active, passive;
           holdtime: infinite, hello interval: 10000 ms
        Addresses bound to peer LDP Ident:
          44.44.44.44     43.44.43.44     104.11.1.2      50.255.102.3
          104.1.34.3      104.77.1.4      104.88.44.3     104.1.1.11
          104.10.11.3     104.1.1.5
        Peer holdtime: 180000 ms; KA interval: 60000 ms; Peer state: estab
        Clients: Dir Adj Client
        LDP Session Protection enabled, state: Ready
            duration: 86400 seconds
```

If the second last line of the output shows `Incomplete`, the Targeted Hello Adjacency is not up yet.

What to do next

These are some troubleshooting steps for MPLS LDP session protection feature:

- Use the **clear mpls ldp neighbor** command if you need to terminate an LDP session after a link goes down. This is useful for situations where the link needs to be taken out of service or needs to be connected to a different neighbor.
- To enable the display of events related to MPLS LDP session protection, use the **debug mpls ldp session protection** command.



CHAPTER 7

MPLS LDP IGP Synchronization

The MPLS LDP IGP synchronization feature ensures that the Label Distribution Protocol (LDP) is fully established before the Interior Gateway Protocol (IGP) path is used for switching. This module explains the concepts related to MPLS LDP IGP synchronization and describes how to configure MPLS LDP IGP synchronization in a network.

Feature history for MPLS LDP IGP synchronization

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS LDP IGP synchronization	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS LDP IGP synchronization

MPLS LDP IGP synchronization is an MPLS LDP synchronization feature that

- provides a means to synchronize LDP and Interior Gateway Protocol (IGP) to minimize MPLS packet loss, and
- ensures that LDP is fully established before the IGP path is used for switching.

MPLS LDP IGP synchronization with peers

When the MPLS LDP IGP synchronization feature is enabled on an interface, the LDP determines if any peer connected by the interface is reachable by looking up the transport address of the peer in the routing table. If a routing entry including longest match or default routing entry for the peer exists, LDP assumes that LDP IGP synchronization is required for the interface and notifies the IGP to wait for LDP convergence.

LDP IGP synchronization with peers requires that the routing table be accurate for the transport address of the peer. If the routing table shows that there is a route for the transport address of the peer, that route must be able to reach that address. However, if the route is a summary route, a default route, or a statically configured route, it may not be the correct route for the peer. You must verify that the route in the routing table can reach the transport address of the peer.

When the routing table has an inaccurate route for the transport address of the peer, LDP cannot set up a session with the peer, which causes the IGP to wait for LDP convergence unnecessarily for the sync hold-down time.

MPLS LDP IGP synchronization incompatibility with IGP nonstop forwarding

The device does not support the MPLS LDP IGP synchronization feature during the startup period if the IGP nonstop forwarding (NSF) is configured. The feature conflicts with IGP NSF when the IGP is performing NSF during startup. After the NSF startup is complete, the device supports MPLS LDP IGP synchronization.

MPLS LDP IGP synchronization compatibility with LDP graceful restart

LDP graceful restart protects traffic when an LDP session is lost. If an interface that supports a graceful restart-enabled LDP session fails, MPLS LDP IGP synchronization is still achieved on the interface while it is protected by graceful restart. MPLS LDP IGP synchronization is eventually lost under these scenarios:

- If LDP fails to restart before the LDP graceful restart reconnect timer expires.
- If an LDP session restarts through other interfaces, but the LDP session on the protected interface fails to recover when the LDP graceful restart recovery timer expires.

MPLS LDP IGP synchronization delay timer

The MPLS LDP IGP synchronization delay timer is a configurable option that:

- allows you to specify a delay time for LDP and IGP synchronization on an interface-by-interface basis
- helps mitigate issues when inaccurate routes cause LDP to wait unnecessarily for convergence, and
- ensures LDP checks synchronization validity after the configured delay before notifying OSPF.

When LDP is fully established and synchronized, it checks the delay timer:

- If you configured a delay time, LDP starts the timer. When the timer expires, LDP checks that synchronization is still valid and notifies the OSPF process.
- If you did not configure a delay time, if synchronization is disabled or down, or if an interface was removed from an IGP process, LDP stops the timer and immediately notifies the OSPF process.
- If you configure a new delay time while a timer is running, LDP saves the new delay time but does not reconfigure the running timer.

How MPLS LDP IGP synchronization works

Summary

These are the key components involved in MPLS LDP IGP synchronization:

- Interior Gateway Protocol (IGP) : Establishes network adjacency and determines optimal paths.
- Label Distribution Protocol (LDP) : Exchanges labels between peers for MPLS forwarding.
- Routing table : Stores network reachability information used by LDP to determine peer reachability.

This process coordinates IGP and LDP actions to ensure that traffic is only forwarded over links where LDP sessions are fully established.

Workflow

These stages describe how MPLS LDP IGP synchronization operates:

1. Synchronization initiation: LDP determines if a peer connected by an interface is reachable by looking up the transport address of the peer in the routing table.
2. IGP notification: If a routing entry for the peer exists, LDP assumes LDP IGP synchronization is required and notifies the IGP to wait for LDP convergence.
3. IGP waiting: The IGP waits for synchronization to be achieved.

   By default, this wait is indefinite, but it can be limited by configuring the **mpls ldp igp sync holddown** command.
4. Max-metric advertisement: If an IGP adjacency is established but LDP IGP synchronization is not yet achieved or is lost, the IGP advertises the max-metric on that link.
5. Synchronization achievement: When LDP is fully established and synchronized, it notifies the IGP process, for example, OSPF.

Result

MPLS LDP IGP synchronization ensures that traffic forwarding only occurs over links where LDP sessions are fully established, thereby preventing packet loss due to unsynchronized protocol states.

Restrictions for MPLS LDP IGP synchronization

The support for MPLS LDP IGP synchronization feature is subjected to these restrictions:

- The feature is supported only on interfaces running Open Shortest Path First (OSPF) or Intermediate System-to-System (IS-IS) processes.
- The feature is not supported on tunnel interfaces or LC-ATM interfaces.
- The feature is not supported with interface-local label space or downstream-on-demand (DoD) requests.
- The feature does not support targeted LDP sessions. Therefore, Any Transport over MPLS (AToM) sessions are not supported.

- The feature does not support Tag Distribution Protocol (TDP). You must specify that the default label distribution protocol is LDP for a device or for an interface.

Configure MPLS LDP IGP synchronization

Configure MPLS LDP IGP synchronization on OSPF interfaces

This task enables MPLS LDP IGP synchronization globally on all interfaces belonging to an OSPF process and specifies LDP as the label distribution protocol.

Procedure

Step 1 Globally enable MPLS hop-by-hop forwarding and set LDP as the label protocol for all interfaces.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
Device(config)# mpls label protocol ldp
```

Step 2 Enable hop-by-hop forwarding on an interface.

Example:

```
Device(config)# interface HundredGigE 1/0/1
Device(config-if)# ip address 10.0.0.11 255.255.255.255
Device(config-if)# mpls ip
Device(config-if)# exit
```

Step 3 Enable OSPF routing on the device.

Example:

```
Device(config)# router ospf 1
Device(config-router)# network 10.0.0.0 0.0.255.255 area 3
```

Step 4 Enable MPLS IGP synchronization on interfaces belonging to an OSPF process.

Example:

```
Device(config-router)# mpls ldp sync
Device(config-router)# end
```

When you issue the **mpls ldp sync** command, all the interfaces that belong to an OSPF process are enabled for MPLS LDP IGP synchronization. To remove LDP IGP synchronization from some interfaces, use the **no mpls ldp igp sync** command on those interfaces.

For example:

```
Device(config)# interface HundredGigE 1/0/1
```

```
Device(config-if)# no mpls ldp igp sync
Device(config-if)# end
```

Step 5 Verify MPLS LDP IGP synchronization with OSPF.

a) Verify if MPLS LDP IGP synchronization is enabled on the interface.

Example:

```
Device# show mpls ldp igp sync

HundredGigE 1/0/1:
 LDP configured;  SYNC enabled.
 SYNC status: sync achieved; peer reachable.
 IGP holddown time: infinite.
 Peer LDP Ident: 10.0.0.1:0
 IGP enabled: OSPF 1
```

The output shows that LDP is configured and that synchronization is enabled.

If MPLS LDP IGP synchronization is not enabled on an interface, the output appears as follows:

```
HundredGigE 1/0/3:
 LDP configured; LDP-IGP Synchronization not enabled.
```

b) Verify the status and configuration of MPLS LDP synchronization with OSPF on specific interfaces.

Example:

```
Device# show ip ospf mpls ldp interface

HundredGigE 1/0/1
  Process ID 1, Area 0
  LDP is configured through LDP autoconfig
  LDP-IGP Synchronization: Yes
  Holddown timer is not configured
  Timer is not running
HundredGigE 1/0/2
  Process ID 1, Area 0
  LDP is configured through LDP autoconfig
  LDP-IGP Synchronization: Yes
  Holddown timer is not configured
  Timer is not running
```

The `LDP-IGP Synchronization` status in the output shows as `Yes`.

Configure MPLS LDP IGP synchronization on an IS-IS interface

This task configures IS-IS on a particular interface and then enables LDP IGP synchronization for IS-IS processes, which then applies to the configured interface.

Procedure

Step 1 Configure the interface to run the IS-IS protocol.

Example:

```
Device(config)# interface HundredGigE 1/0/1
Device(config-if)# ip address 10.50.72.4 255.0.0.0
Device(config-if)# ip router isis
Device(config-if)# exit
```

Step 2 Enable IS-IS routing on the device.

Example:

```
Device(config)# router isis
```

Step 3 Enable MPLS LDP IGP synchronization for all the interfaces belonging to an IS-IS process.

Example:

```
Device(config-router)# mpls ldp sync
Device(config-router)# end
```

Configure MPLS LDP IGP synchronization for all IS-IS interfaces

This task sets up the necessary LDP and IS-IS configurations and enables MPLS LDP IGP synchronization globally for all interfaces belonging to an IS-IS process on the device.

Before you begin

You must define a valid holddown timer for IS-IS.

Procedure

Step 1 Globally enable MPLS hop-by-hop forwarding and set LDP as the label protocol for all interfaces.

Example:

```
Device> enable
Device# configure terminal
Device(config)# mpls ip
Device(config)# mpls label protocol ldp
```

Step 2 Enable IS-IS routing on the device.

Example:

```
Device(config)# router isis ISIS
```

Step 3 Enable MPLS LDP IGP synchronization for all the interfaces belonging to an IS-IS process.

Example:

```
Device(config-router)# mpls ldp sync
Device(config-router)# end
```

Step 4 Enable IS-IS on the required interface.

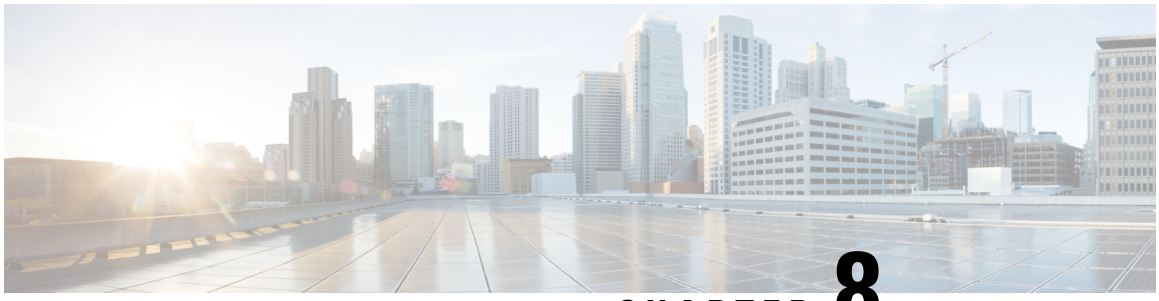
Example:

```
Device(config)# interface HundredGigE 1/0/1
Device(config-if)# ip address 10.25.25.11 255.255.255.0
Device(config-if)# ip router isis ISIS
Device(config-if)# end
```

When you issue the **mpls ldp sync** command, all the interfaces that belong to an IS-IS process are enabled for MPLS LDP IGP synchronization. To remove LDP IGP synchronization from some interfaces, use the **no mpls ldp igp sync** command on those interfaces.

For example:

```
Device(config)# interface HundredGigE 1/0/1
Device(config-if)# no mpls ldp igp sync
Device(config-if)# end
```



CHAPTER 8

MPLS LDP Inbound Label Binding Filtering

MPLS LDP inbound label binding filtering feature enables the configuration of access control lists (ACLs) for controlling the label bindings a label switch router (LSR) accepts from its peer LSRs. This module explains the concepts related to MPLS LDP inbound label binding filtering and describes how to configure this feature on Cisco Smart Switches.

Feature history for MPLS LDP inbound label binding filtering

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS LDP inbound label binding filtering	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS LDP inbound label binding filtering

MPLS LDP inbound label binding filtering is an MPLS LDP feature that

- allows you to configure access control lists (ACLs) for controlling the label bindings a label switch router (LSR) accepts from its peer LSRs.
- helps to manage the memory usage on the switch for storing LDP label bindings advertised by other devices.

For example, in a simple MPLS Virtual Private Network (VPN) environment, the VPN provider edge (PE) devices might require label switched paths (LSPs) only to their peer PE devices. and not to core devices. Inbound label binding filtering enables a PE device to accept labels only from other PE devices.

A label is an identifier that

- is short and fixed-length
- tells switching nodes how to forward data—packets or cells, and
- is used in MPLS for efficient packet forwarding.

A label binding is an association that

- links a destination prefix with a label, and
- is used by LDP to manage label switched paths.

Restriction for MPLS LDP inbound label binding filtering

MPLS LDP inbound label binding filtering supports only standard access control lists (ACLs); it does not support extended ACLs.

Configure MPLS LDP inbound label binding filtering

Use this procedure to configure a device to filter inbound label bindings from LDP neighbors.

This task allows you to control which label bindings an LSR accepts from its peer LSRs, helping to manage memory and network traffic.

Procedure

Step 1 Define a standard IP access list and specify one or more permitted prefixes.

Example:

```
Device> enable
Device# configure terminal
Device(config)# ip access-list standard 1
Device(config-std-nacl)# permit 10.0.0.0
Device(config-std-nacl)# exit
Device(config)#
```

Step 2 Configure MPLS LDP neighbor and specify the ACL to be used for filtering label bindings from that LDP neighbor.

Example:

```
Device# mpls ldp neighbor 10.12.12.12 labels accept 1
Device(config)# end
```

Step 3 Verify MPLS LDP inbound label filtering configuration.

a) Display the status of the LDP session, including the name or number of the ACL configured for inbound filtering.

Example:

```
Device# show mpls ldp neighbor 10.12.12.12 detail
```

```
 Peer LDP Ident: 10.12.12.12:0; Local LDP Ident 10.13.13.13:0
   TCP connection: 10.12.12.12.646 - 10.13.13.13.12592
   State: Oper; Msgs sent/rcvd: 49/45; Downstream; Last TIB rev sent 1257
   Up time: 00:32:41; UID: 1015; Peer Id 0;
   LDP discovery sources:
    Serial1/0/0; Src IP addr: 192.168.1.1
     holdtime: 15000 ms, hello interval: 5000 ms
   Addresses bound to peer LDP Ident:
    10.0.0.129      10.12.12.12     192.168.1.1
   Peer holdtime: 180000 ms; KA interval: 60000 ms; Peer state: estab
   LDP inbound filtering accept acl: 1
```

To display the information about inbound label binding filtering, you must enter the **detail** keyword.

b) Display the contents of current IP access lists or of a specified access list.

Example:

```
Device# show ip access 1
Standard IP access list 1
  permit 10.0.0.0, wildcard bits 0.0.0.255 (1 match)
```

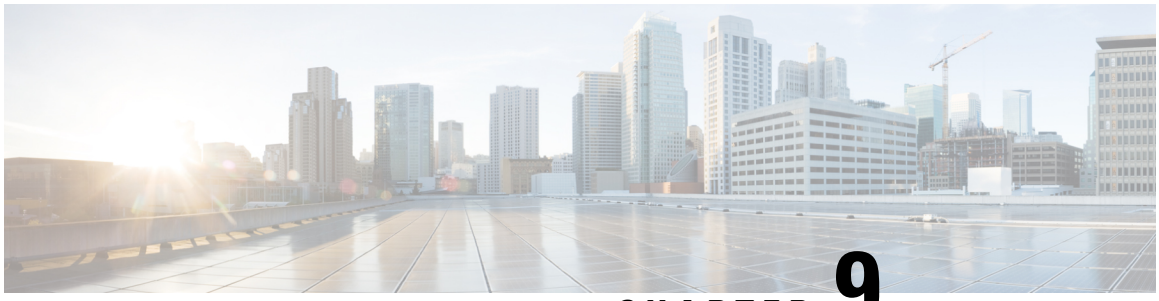
It is important that you enter this command to see how the access list is defined; otherwise, you cannot verify inbound label binding filtering.

Step 4 Verify that the LSR has remote bindings only from a specified peer for prefixes permitted by the access list.

Example:

```
Device# show mpls ldp bindings
 tib entry: 10.0.0.0/8, rev 4
     local binding:  tag: imp-null
 tib entry: 10.2.0.0/16, rev 1137
     local binding:  tag: 16
 tib entry: 10.2.0.0/16, rev 1139
     local binding:  tag: 17
 tib entry: 10.12.12.12/32, rev 1257
     local binding:  tag: 18
 tib entry: 10.13.13.13/32, rev 14
     local binding:  tag: imp-null
 tib entry: 10.10.0.0/16, rev 711
     local binding:  tag: imp-null
 tib entry: 10.0.0.0/8, rev 1135
     local binding:  tag: imp-null
     remote binding: tsr: 10.12.12.12:0, tag: imp-null
 tib entry: 10.0.0.0/8, rev 8
     local binding:  tag: imp-null
```

The device is configured to accept only the label for the specified prefix from the LDP neighbor device, as per the example configuration.



CHAPTER 9

MPLS LDP Local Label Allocation Filtering

MPLS LDP local label allocation filtering feature enables the configuration of filtering policies for selective local label binding assignments by LDP to improve LDP scalability and convergence. This module explains the concepts related to MPLS LDP local label allocation filtering and describes how to configure this feature on Cisco Smart Switches.

Feature history for MPLS LDP inbound label binding filtering

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS LDP inbound label binding filtering	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS LDP local label allocation filtering

MPLS LDP local label allocation filtering is an MPLS LDP feature that

- enables the configuration of filtering policies for selective local label binding assignments by LDP, and
- improves LDP scalability and convergence by reducing the number of local labels allocated and therefore the number of messages exchanged with peers.

Comparison of default LDP local label allocation behavior and LDP behavior with local label allocation controls

In most Layer 3 Virtual Private Network (VPN) configurations only the label switched paths (LSPs) created to reach the /32 host routes or Border Gateway Protocol (BGP) next hops between the provider edge (PE) devices carry traffic and are relevant to the Layer 3 VPNs. LSPs between the PE devices that are not members of a VPN use more memory and create additional processing in LDP across the core. Controlling the local label allocation could off-load LDP processing of non-VPN LSPs in the service provider network core devices.

Figure 5: Default LDP label allocation behavior

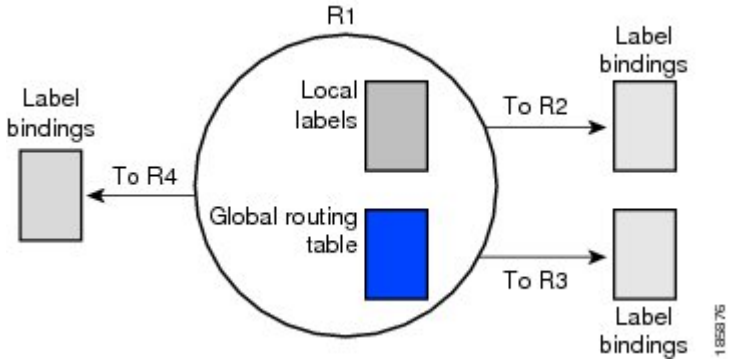
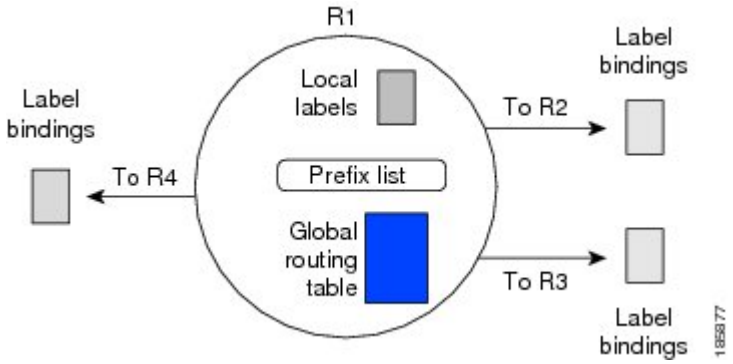


Figure 6: LDP behavior with local label allocation control configured



The figure shows that device R1 learns a number of routes from its IGP neighbors on devices R2, R3, and R4. A prefix list defined on device R1 specifies the prefixes for which LDP allocates a local label.

The table contrasts the default behavior of LDP local label allocation with the behavior when local label allocation controls are configured

Aspect	Default LDP label allocation behavior	LDP behavior with local label allocation control configured
LDP label allocation	The LDP allocates a local label for every route learned from the Interior Gateway Protocol (IGP).	You can configure LDP to selectively allocate local labels for a subset of the prefixes learned from the IGP.

Aspect	Default LDP label allocation behavior	LDP behavior with local label allocation control configured
Label advertisement	The local labels are advertised to and learned by all peers.	The size of the local label space and the number of label binding advertisements are reduced through the use of a prefix list.
Resource usage	Default LDP label allocation uses more memory and create additional processing in LDP across the core.	The decrease in the number of local labels and label binding advertisement messages reduces the amount of memory use and improves convergence time for LDP.

LDP local label filtering and BGP routes

The default behavior of LDP is to allocate local labels for all non-Border Gateway Protocol (BGP) prefixes.

LDP does not apply the configured local label filter to redistributed BGP routes in the global table for which BGP allocates local label, but LDP does the advertisements using Inter-AS Option C. LDP neither forwards these entries nor releases the local labels allocated by BGP.

Benefits of using prefix lists for LDP local label allocation filtering

The MPLS LDP local label allocation filtering feature allows you to configure the LDP to allocate local labels for a subset of the learned prefixes. LDP accepts the prefix and allocates a local label if the prefix is permitted by a prefix list. If the prefix list is not defined, LDP accepts all prefixes and allocates local labels based on its default mode of operation.

Using prefix lists for LDP local label allocation filtering provides these benefits:

- Prefix lists provide more flexibility for specifying a subset of prefixes and masks.
- Prefix lists use a tree-based matching technique which is more efficient than evaluating prefixes or host routes sequentially.
- Prefix lists are easy to modify.

Bahavior change for LDP local label allocation

The MPLS LDP local label allocation filtering feature modifies the local label allocation handling of LDP. The feature supports local label allocation filtering through the specification of a prefix list or host routes.

With the introduction of this feature, LDP needs to determine whether a prefix filter is already configured to control the local label allocation on the local node. If a prefix list exists, the local label allocation is confined to the list of prefixes permitted by the configured prefix list.

LDP also needs to respond to local label allocation configuration changes and to configuration changes that affect the prefix list that LDP is using.

Any of these configuration changes can trigger LDP actions:

- Creating a local label allocation configuration

- Deleting or changing a local label allocation configuration
- Creating a new prefix list for a local label allocation configuration, or
- Deleting or changing a prefix list for a local label allocation configuration.

LDP responds to local label allocation configuration changes by updating the Label Information Database (LIB) and the forwarding table in the global routing table. To update the LIB after a local label filter configuration change without a session reset, LDP keeps all remote bindings.

If you create a local label allocation configuration without defining a prefix list, no LDP action is required. The local label allocation configuration has no effect because the prefix list is created and permits all prefixes.

If you create or change a prefix list and prefixes that were previously allowed are rejected, LDP goes through a label withdraw and release procedure before the local labels for these prefixes are deallocated.

If you delete a prefix, LDP goes through the label withdraw and release procedure for the LIB local label. If the associated prefix is one for which no LIB entry should be allocated, LDP bypasses this procedure.

The LDP default behavior is to allocate local labels for all non-BGP prefixes. This default behavior does not change with the introduction of this feature and the mpls ldp label and **allocate** commands.

Topic 2.1

Restrictions for MPLS LDP local label allocation filtering

MPLS LDP local label allocation filtering is subjected to these restrictions:

- The feature supports prefix lists; it does not support access lists.
- The configuration for prefix list or host routes is supported only in the global routing table.
- LDP and Routing Information Base (RIB) restart handling do not apply.
- The feature does not support Wildcard Forwarding Equalence Class (FEC) requests.
- Remote bindings are retained for LDP table entries that are filtered.

Configure MPLS LDP local label allocation filtering

Use this proecdure to configure filtering policies for selective local label binding assignments by LDP.

This task improves LDP scalability and convergence by controlling which local labels are allocated. You can use either a prefix list or host routes as a filter.

The MPLS LDP local label allocation filtering feature introduces the **mpls ldp label** and **allocate** commands that allow you to configure LDP to selectively allocate local labels for a subset of the prefixes learned from the IGP.

Note A maximum of one local label allocation filter is supported for the global table.

Procedure

Step 1 Create a prefix list for MPLS LDP local label allocation filtering.

Example:

```
Device> enable
Device# configure terminal
Device(config)# ip prefix-list list1 permit 192.168.0.0/16 le 20
```

Specify the input parameters as applicable:

- *list-name*: configures a name to identify the prefix list.
- *list-number*: configures a number to identify the prefix list.
- **seq**: applies a sequence number to a prefix-list entry. The range of sequence numbers is 1 to 4294967294. If a sequence number is not entered when this command is configured, a default sequence numbering is applied to the prefix list. The number 5 is applied to the first prefix entry, and subsequent unnumbered entries are incremented by 5.
- **deny**: denies access for a matching condition.
- **permit**: permits access for a matching condition.
- **network/length**: configures the network address, and the length of the network mask in bits. The network number can be any valid IP address or prefix. The bit mask can be a number from 0 to 32.
- **ge** *ge-length*: represents the greater than or equal to operator along with the minimum prefix length to be matched.
- **le** *le-length*: represents the less than or equal to operator along with the maximum prefix length to be matched

Step 2 Create a prefix list for MPLS LDP local label allocation filtering.

Example:

```
Device(config)# mpls ldp label
```

Step 3 Configure local label allocation filters for learned routes for LDP using a prefix list.

Example:

```
Device(config-ldp-lbl)#allocate global prefix-list list1
```

Specify the input parameters as applicable:

- **global**: specifies the global routing.
- **prefix-list**: specifies a prefix list to be used as a filter for MPLS LDP local label allocation.
- *list-name*: indicates a name that identifies the prefix list.
- *list-number*: indicates a number that identifies the prefix list.

Step 4 Configure local label allocation filters for learned routes for LDP.

Example:

```
Device(config-ldp-lbl)#allocate global host-routes
```

The **host-routes** keyword specifies that local label allocation be done for host routes only.

You can specify that LDP allocate local labels for prefixes configured in a prefix list in the global table or for host routes in the global table.

Step 5 Remove the specific MPLS LDP local label allocation filter without resetting the LDP session..

Example:

```
Device(config-ldp-lbl)#no allocate global host-routes
```

The **host-routes** keyword specifies that host routes be used as a filter for MPLS LDP local label allocation.

Step 6 Remove all local label allocation filters configured under the MPLS LDP label configuration mode and restore LDP default behavior for local label allocation without a session reset.

Example:

```
Device(config-ldp-lbl)#no mpls ldp label
Device(config)#end
```

Step 7 Verify MPLS LDP local label allocation filtering configuration.

a) Verify that local label allocation filtering is configured as expected.

Example:

```
Device# show mpls ldp bindings detail

Advertisement spec:
        Prefix acl = bar
Local label filtering spec: host routes.
  lib entry: 10.1.1.1/32, rev 9
  lib entry: 10.10.7.0/24, rev 10
  lib entry: 10.10.8.0/24, rev 11
  lib entry: 10.10.9.0/24, rev 12
  lib entry: 10.41.41.41/32, rev 17
  lib entry: 10.50.50.50/32, rev 15
  lib entry: 10.60.60.60/32, rev 18
  lib entry: 10.70.70.70/32, rev 16
  lib entry: 10.80.80.80/32, rev 14
```

The output of this command verifies that host routes are configured as the local label allocation filter for the device.

b) Verify that local label allocation filtering was configured properly and display how LDP accepts or withdraws labels.

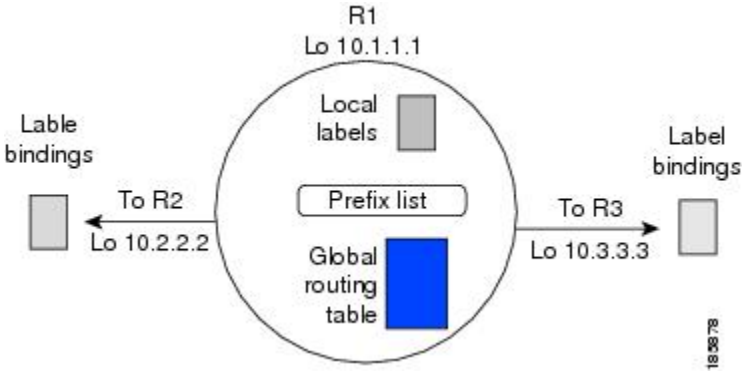
Example:

```
Device# debug mpls ldp binding filter
LDP Local Label Allocation Filtering changes debugging is on
.
.
.
```

Sample configuration for MPLS LDP local label filtering

This section provides a sample configuration for MPLS LDP local label allocation filtering.

Figure 7: MPLS LDP local label allocation filtering example



In this sample configuration:

- Devices R1, R2, and R3 have loopback addresses 10.1.1.1, 10.2.2.2, and 10.3.3.3 defined and advertised by the IGP, respectively.
- 10.1.1.1 is the router ID of device R1, 10.2.2.2 is the router ID of device R2, and 10.3.3.3 is the router ID of device R3.
- A prefix list is defined on device R1 to specify the local labels for which LDP allocates a local label.

You can use the LDP CLI commands to verify that

- device R1 has allocated a local label for the correct subset of the prefixes, and
- devices R2 and R3 did not receive any remote bindings for the prefixes for which device R1 did not assign a local label.

Routing table on device R1

Enter the **show ip route** command to display the current state of the routing table. This example shows the routing table on device R1 based on the sample configuration.

```
Device# show ip route

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
     10.0.0.0/32 is subnetted, 1 subnets
C       10.1.1.1 is directly connected, Loopback0
     10.2.0.0/32 is subnetted, 1 subnets
O       10.2.2.2 [110/11] via 10.10.7.1, 00:00:36, FastEthernet1/0/0
     10.3.0.0/32 is subnetted, 1 subnets
O       10.3.3.3 [110/11] via 10.10.9.1, 00:00:36, FastEthernet3/0/0
     10.0.0.0/24 is subnetted, 3 subnets
```

```
C       10.10.7.0 is directly connected, FastEthernet1/0/0
O       10.10.8.0 [110/20] via 10.10.9.1, 00:00:36, FastEthernet3/0/0
                  [110/20] via 10.10.7.1, 00:00:36, FastEthernet1/0/0
C       10.10.9.0 is directly connected, FastEthernet3/0/0
```

Local label bindings on devices R1, R2, and R3

Enter the **show mpls ldp bindings** command on devices R1, R2, and R3 to display the contents of the Label Information Base (LIB) on each device. In these examples, the default LDP allocation behavior is in operation; that is, LDP allocates a local label for every route and advertises a label binding for every route learned from the IGP.

LIB on Device R1:

```
Device# show mpls ldp bindings

  lib entry: 10.1.1.1/32, rev 7
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 16
        remote binding: lsr: 10.2.2.2:0, label: 17
  lib entry: 10.2.2.2/32, rev 13
        local binding:  label: 1000
        remote binding: lsr: 10.3.3.3:0, label: 18
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.3.3.3/32, rev 15
        local binding:  label: 1002
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 18
  lib entry: 10.10.7.0/24, rev 8
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 17
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.10.8.0/24, rev 11
        local binding:  label: 1001
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.10.9.0/24, rev 9
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 16
```

The local labels assigned to 10.2.2.2 and 10.3.3.3 on device R1 are advertised to devices R2 and R3.

LIB on device R2:

```
Device# show mpls ldp bindings

  lib entry: 10.1.1.1/32, rev 11
        local binding:  label: 17
        remote binding: lsr: 10.3.3.3:0, label: 16
        remote binding: lsr: 10.1.1.1:0, label: imp-null
  lib entry: 10.2.2.2/32, rev 7
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 18
        remote binding: lsr: 10.1.1.1:0, label: 1000
  lib entry: 10.3.3.3/32, rev 15
        local binding:  label: 18
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: 1002
  lib entry: 10.10.7.0/24, rev 8
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 17
```

```
        remote binding: lsr: 10.1.1.1:0, label: imp-null
  lib entry: 10.10.8.0/24, rev 9
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: 1001
  lib entry: 10.10.9.0/24, rev 13
        local binding:  label: 16
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: imp-null
```

LIB on device R3:

```
Device# show mpls ldp bindings
  lib entry: 10.1.1.1/32, rev 13
        local binding:  label: 16
        remote binding: lsr: 10.2.2.2:0, label: 17
        remote binding: lsr: 10.1.1.1:0, label: imp-null
  lib entry: 10.2.2.2/32, rev 15
        local binding:  label: 18
        remote binding: lsr: 10.2.2.2:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: 1000
  lib entry: 10.3.3.3/32, rev 7
        local binding:  label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 18
        remote binding: lsr: 10.1.1.1:0, label: 1002
  lib entry: 10.10.7.0/24, rev 11
        local binding:  label: 17
        remote binding: lsr: 10.2.2.2:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: imp-null
  lib entry: 10.10.8.0/24, rev 8
        local binding:  label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: 1001
  lib entry: 10.10.9.0/24, rev 9
        local binding:  label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 16
        remote binding: lsr: 10.1.1.1:0, label: imp-null
```

Local label allocation filtering configuration on device R1

Enter the **mpls ldp label** command to configure a local label allocation filter. These examples show how to configure a local label allocation filter by host routes only and by a prefix list.

- Local label allocation filter—host routes only configuration:

  This example shows the selection of host routes as the only filter. Here, the local label allocation filtering is defined on device R1 under MPLS LDP label configuration mode

  ```
  configure terminal
  !
  mpls ldp label
   allocate global host-routes
   exit
  exit
  ```

- Local label allocation filter—prefix list configuration:

  This example shows how to configure a local label allocation filter that allows or denies prefixes based on a prefix list:

  ```
  configure terminal
  ```

```
!
mpls ldp label
 allocate global prefix-list ListA
 exit
end
```

ListA is a prefix list defined as:

```
configure terminal
!
ip prefix-list ListA permit 0.0.0.0/32 ge 32
```

Local label allocation filtering changes label bindings on devices R1, R2, and R3

After configuring a local label allocation filter on Device R1, enter the **show mpls ldp bindings** command again to see the changes in the local label bindings in the LIB on each device. Changes to the output in the LIB entries are highlighted in bold text.

- LIB on device R1 after local label allocation filtering

  This example shows how the configuration of a local label allocation prefix-list filter changes the contents of the LIB on device R1.

```
Device# show mpls ldp bindings

  lib entry: 10.1.1.1/32, rev 7
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 16
        remote binding: lsr: 10.2.2.2:0, label: 17
  lib entry: 10.2.2.2/32, rev 13
        local binding:  label: 1000
        remote binding: lsr: 10.3.3.3:0, label: 18
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.3.3.3/32, rev 15
        local binding:  label: 1002
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 18
  lib entry: 10.10.7.0/24, rev 8
        no local binding
        remote binding: lsr: 10.3.3.3:0, label: 17
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.10.8.0/24, rev 11
        no local binding
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.10.9.0/24, rev 9
        no local binding
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 16
```

  Local label bindings for all but 10.2.2.2 and 10.3.3.3 on device R1 are advertised as withdrawn.

- LIB on device R2 after local label allocation filtering

  This example shows how the configuration of a local label allocation prefix-list filter on device R1 changes the contents of the LIB on device R2.

```
Device# show mpls ldp bindings
  lib entry: 10.1.1.1/32, rev 11
        local binding:  label: 17
        remote binding: lsr: 10.3.3.3:0, label: 16
```

```
  lib entry: 10.2.2.2/32, rev 7
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 18
        remote binding: lsr: 10.1.1.1:0, label: 1000
  lib entry: 10.3.3.3/32, rev 15
        local binding:  label: 18
        remote binding: lsr: 10.3.3.3:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: 1002
lib entry: 10.10.7.0/24, rev 8
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: 17
  lib entry: 10.10.8.0/24, rev 9
        local binding:  label: imp-null
        remote binding: lsr: 10.3.3.3:0, label: imp-null
  lib entry: 10.10.9.0/24, rev 13
        local binding:  label: 16
        remote binding: lsr: 10.3.3.3:0, label: imp-null
```

The 10.10.7.0/24, 10.10.8.0/24, and 10.10.9.0/24 prefixes are no longer assigned local labels. Therefore, device R1 sends no label advertisement for these prefixes.

- LIB on device R3 after local label allocation filtering

  This example shows how the configuration of a local label allocation prefix-list filter on device R1 changes the contents of the LIB on device R3.

```
Device# show mpls ldp bindings
  lib entry: 10.1.1.1/32, rev 13
        local binding:  label: 16
        remote binding: lsr: 10.2.2.2:0, label: 17
        remote binding: lsr: 10.1.1.1:0, label: imp-null
  lib entry: 10.2.2.2/32, rev 15
        local binding:  label: 18
        remote binding: lsr: 10.2.2.2:0, label: imp-null
        remote binding: lsr: 10.1.1.1:0, label: 1000
  lib entry: 10.3.3.3/32, rev 7
        local binding:  label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 18
        remote binding: lsr: 10.1.1.1:0, label: 1002
  lib entry: 10.10.7.0/24, rev 11
        local binding:  label: 17
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.10.8.0/24, rev 8
        local binding:  label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: imp-null
  lib entry: 10.10.9.0/24, rev 9
        local binding:  label: imp-null
        remote binding: lsr: 10.2.2.2:0, label: 16
```

The 10.10.7.0/24, 10.10.8.0/24, and 10.10.9.0/24 prefixes are no longer assigned local labels. Again, device R1 sends no label advertisement for these prefixes.

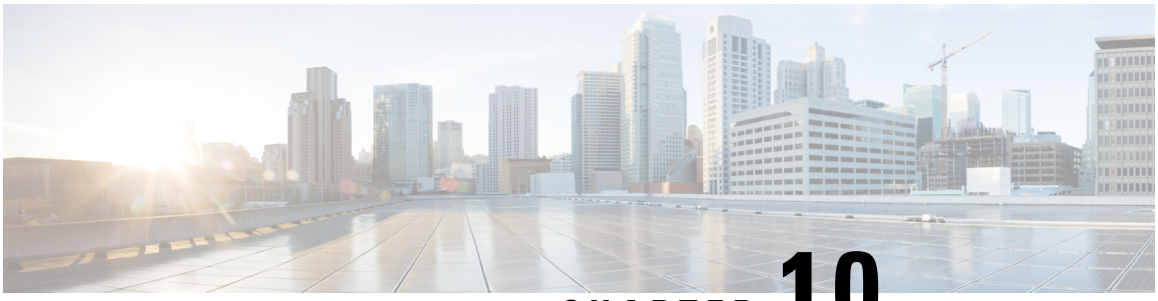
Display the local label allocation filter

Enter the **show mpls ldp detail** command to display the filter used for local label allocation. For example:

```
Device# show mpls ldp bindings detail

Advertisement spec:
        Prefix acl = List1
Local label filtering spec: host routes. ! <--- Local local label filtering spec
```

```
lib entry: 10.1.1.1/32, rev 9
lib entry: 10.10.7.0/24, rev 10
lib entry: 10.10.8.0/24, rev 11
lib entry: 10.10.9.0/24, rev 12
lib entry: 10.41.41.41/32, rev 17
lib entry: 10.50.50.50/32, rev 15
lib entry: 10.60.60.60/32, rev 18
lib entry: 10.70.70.70/32, rev 16
lib entry: 10.80.80.80/32, rev 14
```



CHAPTER 10

MPLS LDP Graceful Restart

When a router is configured with MPLS LDP graceful restart feature, it assists a neighboring router that has MPLS LDP stateful switchover or nonstop forwarding (SSO or NSF) support and graceful restart to recover gracefully from an interruption in service. This module explains the concepts related to MPLS LDP graceful restart and describes how to configure it on Cisco Smart Switches.

Feature history for MPLS LDP graceful restart

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 17.18.1	MPLS LDP graceful restart	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

MPLS LDP graceful restart

MPLS LDP graceful restart (GR) is a high-availability feature that

- assists a neighboring device that has MPLS LDP stateful switchover (SSO) or nonstop forwarding (NSF) support and graceful restart to recover gracefully from an interruption in service
- enables an SSO- or NSF-enabled device to become operational more quickly by maintaining its forwarding state when the LDP session is interrupted, and
- functions strictly in helper mode, meaning it can only help other devices that are enabled with MPLS SSO or NSF and GR to recover.

How a route processor advertises MPLS LDP graceful restart support

A route processor (RP) that is configured to perform MPLS LDP GR includes the Fault Tolerant (FT) Type Length Value (TLV) in the LDP initialization message. The RP sends the LDP initialization message to a neighbor to establish an LDP session.

The FT session TLV includes this information:

- The Learn from Network (L) flag is set to 1, which indicates that the route processor is configured to perform MPLS LDP GR.
- The Reconnect Timeout field shows the time, in milliseconds, that the neighbor should wait for a reconnection if the LDP session is lost. In this release, the timer is set to 0, which indicates that if the local switch fails, its peers should not wait for it to recover. The timer setting indicates that the local switch is working in helper mode.
- The Recovery Time field shows the time, in milliseconds, that the neighbor should retain the MPLS forwarding state during a recovery. If a neighbor did not preserve the MPLS forwarding state before the restart of the control plane, the neighbor sets the recovery time to 0.

If two route processors establish an LDP session and one route processor is not configured for MPLS LDP GR, the two route processors create a normal LDP session but do not have the ability to perform MPLS LDP GR. Both route processors must be configured for MPLS LDP GR.

How MPLS LDP graceful restart works

Summary

The key components involved in the process are:

- Device configured with MPLS LDP GR (helper device): Assists a peer device in recovering from service interruptions by maintaining stale label bindings and re-establishing LDP sessions.
- Device configured with MPLS LDP SSO or NSF (recovering device): The device that experiences a service disruption such as a TCP/UDP event, route processor switchover, and needs to recover its MPLS forwarding state.

When you enable MPLS LDP GR on a device that peers with an MPLS LDP SSO- or NSF-enabled router, the SSO- or NSF-enabled router can maintain its forwarding state when the LDP session between them is interrupted. While the SSO- or NSF-enabled router recovers, the peer router forwards packets using stale information. This enables the SSO- or NSF-enabled router to become operational more quickly.

Workflow

These stages describe how MPLS LDP graceful restart works:

1. Interruption detection: A helper device notices an interruption in service with the recovering device.
2. Stale binding marking: The helper device marks all label bindings received from the recovering device as stale but continues to use them for MPLS forwarding.
3. Session re-establishment: The helper device re-establishes an LDP session with the recovering device while retaining its stale label bindings.

4. Label binding re-advertisement: Both devices re-advertise their label binding information. If the helper device relearns a label from the recovering device after the session is established, the stale flags are removed.

Restrictions for MPLS LDP graceful restart

The support for MPLS LDP graceful restart feature is subjected to these restrictions:

- MPLS LDP GR is supported only in strict helper mode.
- MPLS LDP GR cannot be configured on label-controlled ATM (LC-ATM) interfaces.

Behavior without MPLS LDP graceful restart configuration

If two route processors establish an LDP session and one route processor is not configured for MPLS LDP GR, the two route processors create a normal LDP session but do not have the ability to perform MPLS LDP GR. Both route processors must be configured for MPLS LDP GR to function.

Configure MPLS LDP graceful restart

This task configures the switch to operate in helper mode. In this mode, the router assists neighboring peers enabled with SSO or NSF to recover gracefully. Enabling this feature globally affects only new LDP sessions.

Before you begin

- Ensure that Cisco Express Forwarding (CEF) is enabled on the switch.
- You must enable MPLS LDP graceful restart GR on all route processors for an LDP session to be preserved during an interruption in service.

Procedure

Step 1 Enable distributed CEF on the switch.

Example:

```
Device> enable
Device# configure terminal
Device(config)# ip cef distributed
```

The **mpls ip** command is enabled by default. You do not have to specify this command.

Globally enabling MPLS forwarding does not enable it on the device interfaces. You must enable MPLS forwarding on the interfaces as well as for the device.

Step 2 Enable the switch to protect LDP bindings and the MPLS forwarding state.

Example:

```
Device(config)# mpls ldp graceful-restart
```

The **mpls ip** command is enabled by default. You do not have to specify this command.

Globally enabling MPLS forwarding does not enable it on the device interfaces. You must enable MPLS forwarding on the interfaces as well as for the device.

Step 3 Configure MPLS hop-by-hop forwarding for an interface.

Example:

```
Device(config)# interface HundredGigE 0/1/0
Device(config-if)# mpls ip
Device(config-if)#mpls label protocol ldp
Device(config-if)# exit
Device(config)#exit
```

You can also issue the **mpls label protocol ldp** command in global configuration mode, which enables LDP on all interfaces configured for MPLS.

Step 4 Verify MPLS LDP graceful restart configuration

a) Display graceful restart information for LDP sessions.

Example:

```
Device# show mpls ldp neighbor graceful-restart

Peer LDP Ident: 10.20.20.20:0; Local LDP Ident 10.17.17.17:0
   TCP connection: 10.20.20.20.16510 - 10.17.17.17.646
    State: Oper; Msgs sent/rcvd: 8/18; Downstream
    Up time: 00:04:39
    Graceful Restart enabled; Peer reconnect time (msecs): 120000
Peer LDP Ident: 10.19.19.19:0; Local LDP Ident 10.17.17.17:0
    TCP connection: 10.19.19.19.11007 - 10.17.17.17.646
    State: Oper; Msgs sent/rcvd: 8/38; Downstream
    Up time: 00:04:30
    Graceful Restart enabled; Peer reconnect time (msecs): 120000
```

b) Display graceful restart sessions and session parameters.

Example:

```
Device# show mpls ldp neighbor graceful-restart
LDP Graceful Restart is enabled
Neighbor Liveness Timer: 120 seconds
Max Recovery Time: 120 seconds
Forwarding State Holding Time: 600 seconds
Down Neighbor Database (0 records):
Graceful Restart-enabled Sessions:
  VRF default:
    Peer LDP Ident: 44.44.44.44:0, State: estab
    Peer LDP Ident: 11.11.11.11:0, State: estab
```

c) Display the status of LDP sessions.

Example:

```
Device# show mpls ldp neighbor
Peer LDP Ident: 10.1.1.2:0; Local LDP Ident 10.1.1.1:0
TCP connection: 10.1.1.2.18 - 10.1.1.1.66
State: Oper; Msgs sent/rcvd: 12/11; Downstream
Up time: 00:00:10
LDP discovery sources:
```

```
HundredGigE0/1/0, Src IP addr: 10.20.10.2
Addresses bound to peer LDP Ident:
10.1.1.2    10.20.20.1    10.20.10.2
```

The device is now configured to support MPLS LDP graceful restart in helper mode. New LDP sessions will include the Fault Tolerant (FT) Type Length Value (TLV) in their initialization messages.