Implementing MPLS Label Distribution Protocol

Implementing MPLS Label Distribution Protocol

MPLS (Multi Protocol Label Switching) is a forwarding mechanism based on label switching. In an MPLS network, data packets are assigned labels and packet-forwarding decisions are taken based on the contents of the label. To switch labeled packets across the MPLS network, predetermined paths are established for various source-destination pairs. These predetermined paths are known as Label Switched Paths (LSPs). To establish LSPs, MPLS signaling protocols are used. Label Distribution Protocol (LDP) is an MPLS signaling protocol used for establishing LSPs. This module provides information about how to configure MPLS LDP.

IPv6 Support in MPLS LDP-MPLS LDPv6 makes the LDP control plane to run on IPv6 in order to setup LSPs for IPv6 prefixes. You can enable MPLS LDPv6 along with existing IPv4 services. LDPv6 feature support is explained in the *IPv6 Support in MPLS LDP* section.

Note In the 7.2.1 release, LDPv6 is only supported for routers in an LSR role, and not the LDP Label Edge Router (LER) role.

Load Balancing with MPLS Traffic

Forwarding to a single destination can occur over multiple best paths, which tie for top place in routing metric calculations. Traffic flows are distributed across the multiple paths using a distribution algorithm, which you

can choose based on the traffic flow pattern. Load balancing decisions are made on ingress line card based on your selected load-balancing profile.

To allow hashing for the Label Edge Router (LER) and Label Switched Routers (LSRs) with MPLS traffic or algorithm to use the inner ethernet fields of the source MAC and destination MAC addresses, use the **hw-module profile load-balance algorithm** command with a suitable load-balancing profile.

You can choose a suitable load-balancing profile (mpls-lsr-ler or mpls-lsr-ler-optimized) based on the prominent traffic flow on the device as recommended in the following table.

Table 1: MPLS Load-Balancing Profiles for Different Traffic Flows

Traffic Flow	Description	Recommended Load-Balancing Profile
4 Label IPv6 flows (EthoMPLS4/6oIPv6)	Flows that have four to six MPLS labels, followed by IPv6 and next header	mpls-lsr-ler-optimized
IPv6 pop and lookup flows (EthoMPLS2/3oIPv6oXX) with L4 as non-TCP/UDP (for example, no next header, GRE)	Flows that have two or three MPLS labels, followed by IPv6 and either no next header or a next header as non-TCP/UDP (for example, GRE, ESP)	mpls-lsr-ler-optimized
IPv4 pop and lookup flows (EthoMPLS2/3oIPv4oL4) with L4 as TCP or UDP	Flows that have two or three MPLS labels, followed by IPv4 and next header as TCP/UDP. **Note** GRE header is parsed along with IPv4 header. Therefore, if GRE options such as key and sequence number are used, IPv4 source IP and destination IP may point to incorrect offsets, resulting in out of order packets.	mpls-lsr-ler
IPv6 pop and lookup flows (EthoMPLS2/3oIPv6oXX) with L4 as TCP or UDP	Flows that have two or three MPLS labels, followed by IPv6 and next header as TCP/UDP	mpls-lsr-ler

Prerequisites for Implementing MPLS Label Distribution Protocol

The following are the prerequisites to implement MPLS LDP:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must be running Cisco IOS XR software.
- You must install a composite mini-image and the MPLS package.

- You must activate IGP.
- We recommend to use a lower session holdtime bandwidth such as neighbors so that a session down occurs before an adjacency-down on a neighbor. Therefore, the following default values for the hello times are listed:
  - Holdtime is 15 seconds.
  - Interval is 5 seconds.

  For example, the LDP session holdtime can be configured as 30 seconds by using the **holdtime** command.

Restrictions for MPLS LDP

- When paths of different technologies are resolved over ECMP, it results in *heterogeneous* ECMP, leading to severe network traffic issues. Don't use ECMP for any combination of the following technologies:
  - LDP.
  - BGP-LU, including services over BGP-LU loopback peering or recursive services at Level-3.
  - VPNv4.
  - 6PE and 6VPE.
  - EVPN.
  - Recursive static routing.

Overview of Label Distribution Protocol

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
BFD, LACP Triggering TE FRR	Release 7.3.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operate in the native mode.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Targeted LDP	Release 7.3.1	This feature is now supported on routers that have Cisco NC57 line cards installed and operate in the native mode.

In IP forwarding, when a packet arrives at a router the router looks at the destination address in the IP header, performs a route lookup, and then forwards the packet to the next hop. MPLS is a forwarding mechanism in which packets are forwarded based on labels. Label Distribution Protocols assign, distribute, and install the labels in an MPLS environment. It is the set of procedures and messages by which Label Switched Routers

(LSRs) establish LSPs through a network by mapping network-layer routing information directly to data-link layer switched paths. These LSPs may have an endpoint at a directly attached neighbor (comparable to IP hop-by-hop forwarding), or may have an endpoint at a network egress node, enabling switching via all intermediary nodes.

LSPs can be created statically, by RSVP traffic engineering (TE), or by LDP. LSPs created by LDP perform hop-by-hop path setup instead of an end-to-end path. LDP enables LSRs to discover their potential peer routers and to establish LDP sessions with those peers to exchange label binding information. Once label bindings are learned, the LDP is ready to set up the MPLS forwarding plane.

For more information about setting up LSPs, see MPLS Label Distribution Protocol : Details, on page 25.

Configuring Label Distribution Protocol

Depending on the requirements, LDP requires some basic configuration tasks described in the following topics:

Configuring Label Distribution Protocol

This section explains the basic LDP configuration. LDP should be enabled on all interfaces that connects the router to potential LDP peer routers. You can enable LDP on an interface by specifying the interface under mpls ldp configuration mode.

Configuration Example

This example shows how to enable LDP over an interface.

```
RP/0/RP0/CPU0:Router(config)#  mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)#  router-id 192.168.70.1
RP/0/RP0/CPU0:Router(config-ldp)#  interface HundredGigE 0/0/0/5
RP/0/RP0/CPU0:Router(config-ldp-if)#  commit
```

Configuring Label Distribution Protocol Discovery Parameters

LSRs that are running LDP send hello messages on all the LDP enabled interfaces to discover each other. So, the LSR that receives the LDP hello message on an interface is aware of the presence of the LDP router on that interface. If LDP hello messages are sent and received on an interface, there's an LDP adjacency across the link between the two LSRs that are running LDP. By default, hello messages are sent every 5 seconds with a hold time of 15 seconds. If the LSR doesn't receive a discovery hello from peer before the hold time expires, the LSR removes the peer LSR from the list of discovered LDP neighbors. The LDP discovery parameters can be configured to change the default parameters.

LDP session between LSRs that aren't directly connected is known as targeted LDP session. For targeted LDP sessions, LDP uses targeted hello messages to discover the extended neighbors. By default, targeted hello messages are sent every 10 seconds with a hold time of 90 seconds.

Configuration Example

This example shows how to configure the following LDP discovery parameters:

- hello hold time

- hello interval
- targeted hello hold time
- targeted hello interval

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# router-id 192.168.70.1
RP/0/RP0/CPU0:Router(config-ldp)# discovery hello holdtime 30
RP/0/RP0/CPU0:Router(config-ldp)# discovery hello interval 10
RP/0/RP0/CPU0:Router(config-ldp)# discovery targeted-hello holdtime 120
RP/0/RP0/CPU0:Router(config-ldp)# discovery targeted-hello interval 15
RP/0/RP0/CPU0:Router(config-ldp)# commit
```

Verification

This section verifies the MPLS LDP discovery parameters configuration.

```
RP/0/RP0/CPU0:Router# show mpls ldp parameters
LDP Parameters:
Role: Active
Protocol Version: 1
Router ID: 192.168.70.1
Discovery:
Link Hellos:     Holdtime:30 sec, Interval:10 sec
Targeted Hellos: Holdtime:120 sec, Interval:15 sec
Quick-start: Enabled (by default)
Transport address:       IPv4: 192.168.70.1
```

Label Distribution Protocol Discovery for Targeted Hellos

LDP session between LSRs that aren't directly connected is known as targeted LDP session. For LDP neighbors which aren't directly connected, you should manually configure the LDP neighborship on both the routers.

Configuration Example

This example shows how to configure LDP for non-directly connected routers, Router 1, and Router 2.

```
RP/0/RP0/CPU0:Router1(config)#  mpls ldp
RP/0/RP0/CPU0:Router1(config-ldp)# router-id 192.168.70.1
RP/0/RP0/CPU0:Router2(config-ldp)# address-family ipv4
RP/0/RP0/CPU0:Router2(config-ldp-af)#discoverey targeted-hello accept
RP/0/RP0/CPU0:Router1(config-ldp-af)# neighbor 172.20.10.10 targeted
RP/0/RP0/CPU0:Router1(config-ldp-af)# interface HundredGigE 0/0/0/5
RP/0/RP0/CPU0:Router1(config-ldp-if)# commit
```

```
RP/0/RP0/CPU0:Router2(config)#  mpls ldp
RP/0/RP0/CPU0:Router2(config-ldp)# router-id 172.20.10.10
RP/0/RP0/CPU0:Router2(config-ldp)# address-family ipv4
RP/0/RP0/CPU0:Router2(config-ldp-af)#discoverey targeted-hello accept
RP/0/RP0/CPU0:Router2(config-ldp-af)# neighbor 192.168.70.1 targeted
RP/0/RP0/CPU0:Router2(config-ldp-af)# commit
```

Enhanced Targeted LDP Session Scale Values

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Enhanced Targeted LDP Session Scale Values	Release 7.6.1	With the unidimensional scale parameter value increased to 1999 on the router, you can now configure more targeted LDP sessions, which use targeted "hello" messages to discover extended neighbors in an MPLS network.

Starting from Cisco IOS XR Release 7.6.1, the targeted LDP session scale value is increased to 1999 on the router. Local Packet Transport Services (LPTS) maintains a table in Ternary content-addressable memory (TCAM), which helps in delivering packets to the intended application on the Routing Processor (RP). Whenever a targeted LDP session is established for an advertised prefix, two LDP entries are pushed into LPTS table corresponding to that prefix.

By default, there is a scale limit for each flow. You need to manually allocate maximum entries to LDP flow by deallocating entries of another flow that are not in use. For a targeted LDP session, two flow type entries are pushed into LPTS: **LDP-TCP-known** and **LDP-TCP-cfg-peer**. For the underlying LDP session with the peer, you need two more LDP entries in the hardware. As a result, you need to support a Unidimensional scale, and you can configure maximum entries of all the other flows.

Configure the maximum entries of a particular flow in hardware by using the following command:

```
lpts pifib hardware dynamic-flows location <loc> flow <flow_name> max <max_entries>
```

In total, you can configure upto 8000 entries for all the flows combined in the hardware. However, the total supported LDP (including targeted LDP and LDP) scale on routers is a maximum of 1999 sessions.

Apart from these flows, you can deallocate maximum entries allocated for other flows that are not in use and allocate them to **LDP-TCP-known** and **LDP-TCP-configured** flow types, which helps in achieving the required scale.

After you configure, use the following commands to see the configured values:

```
show lpts pifib dynamic flow statistics location <loc>
```

For more information on LPTS Flow Types, see chapter *LPTS* of *IP Addresses and Services Configuration Guide*.

Label Advertisement Control

LDP allows you to control the advertising and receiving of labels. You can control the exchange of label binding information by using label advertisement control (outbound filtering) or label acceptance control (inbound filtering).

Label Advertisement Control (Outbound Filtering)

Label Distribution Protocol advertises labels for all the prefixes to all its neighbors. When this is not desirable (for scalability and security reasons), you can configure LDP to perform outbound filtering for local label advertisement for one or more prefixes to one more peers. This feature is known as LDP outbound label filtering, or local label advertisement control. You can control the exchange of label binding information using the **mpls ldp label advertise** command. Using the optional keywords, you can advertise selective prefixes

to all neighbors, advertise selective prefixes to defined neighbors, or disable label advertisement to all peers for all prefixes. Prefixes and peers advertised selectively are defined in the access list.

Configuration Example: Label Advertisement Control

This example shows how to configure outbound label advertisement control. In this example, neighbors are specified to advertise and receive label advertisements. Also an interface is specified for label advertisement.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# address-family ipv4
RP/0/RP0/CPU0:Router(config-ldp-af)# label local advertise to 10.0.0.1:0 for pfx_ac11
RP/0/RP0/CPU0:Router(config-ldp-af)# label local advertise interface TenGigE 0/0/0/5

RP/0/RP0/CPU0:Router(config-ldp-af)# commit
```

Label Acceptance Control (Inbound Filtering)

LDP accepts labels (as remote bindings) for all prefixes from all peers. LDP operates in liberal label retention mode, which instructs LDP to keep remote bindings from all peers for a given prefix. For security reasons, or to conserve memory, you can override this behavior by configuring label binding acceptance for set of prefixes from a given peer. The ability to filter remote bindings for a defined set of prefixes is also referred to as LDP inbound label filtering or label acceptance control.

Configuration Example : Label Acceptance Control (Inbound Filtering)

This example shows how to configure label acceptance control. In this example, an LSR is configured to accept and retain label bindings from neighbors for prefixes defined in access list .

```
RP/0/RP0/CPU0:Router(config)#mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)#address-family ipv4
RP/0/RP0/CPU0:Router(config-ldp-af)#label remote accept from 192.168.1.1:0 for acl_1
RP/0/RP0/CPU0:Router(config-ldp-af)#label remote accept from 192.168.2.2:0 for acl_2
RP/0/RP0/CPU0:Router(config-ldp-af)#commit
```

Configuring Local Label Allocation Control

LDP creates label bindings for all IGP prefixes and receives label bindings for all IGP prefixes from all its peers. If an LSR receives label bindings from several peers for thousands of IGP prefixes, it consumes significant memory and CPU. In some scenarios, most of the LDP label bindings may not useful for any application and you may required to limit the allocation of local labels. This is accomplished using LDP local label allocation control, where an access list can be used to limit allocation of local labels to a set of prefixes. Limiting local label allocation provides several benefits, including reduced memory usage requirements, fewer local forwarding updates, and fewer network and peer updates.

Configuration Example

This example shows how to configure local label allocation using an IP access list to specify a set of prefixes that local labels can allocate and advertise.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# address-family ipv4
RP/0/RP0/CPU0:Router(config-ldp-af)# label local allocate for pfx_acl_1
RP/0/RP0/CPU0:Router(config-ldp-af)# commit
```

Configuring Downstream on Demand

By default, LDP uses downstream unsolicited mode in which label advertisements for all routes are received from all LDP peers. The downstream on demand feature adds support for downstream-on-demand mode, where the label is not advertised to a peer, unless the peer explicitly requests it. At the same time, since the peer does not automatically advertise labels, the label request is sent whenever the next-hop points out to a peer that no remote label has been assigned.

In downstream on demand configuration, an ACL is used to specify the set of peers for downstream on demand mode. For down stream on demand to be enabled, it needs to be configured on both peers of the session. If only one peer in the session has downstream-on-demand feature configured, then the session does not use downstream-on-demand mode.

Configuration Example

This example shows how to configure LDP Downstream on Demand.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# session downstream-on-demand with ACL1
RP/0/RP0/CPU0:Router(config-ldp)# commit
```

Configuring Explicit Null Label

Cisco MPLS LDP uses implicit or explicit null label as local label for routes or prefixes that terminate on the given LSR. These routes include all local, connected, and attached networks. By default, the null label is **implicit-null** that allows LDP control plane to implement penultimate hop popping (PHP) mechanism. When this is not desirable, you can configure **explicit-null** label that allows LDP control plane to implement ultimate hop popping (UHP) mechanism. You can configure explicit-null feature on the ultimate hop LSR. Access-lists can be used to specify the IP prefixes for which PHP is desired.

You can enforce implicit-null local label for a specific prefix by using the **implicit-null-override** command even if the prefix requires a non-null label to be allocated by default. For example, by default, an LSR allocates and advertises a non-null label for an IGP route. If you wish to terminate LSP for this route on penultimate hop of the LSR, you can enforce implicit-null label allocation and advertisement for this prefix using the **implicit-null-override** command.

Note If the outgoing label is *implicit-null* on the penultimate hop (of the label switched path), the outermost label is removed, and the payload forwarded. The payload is accounted as *MPLS* even if it is IP traffic, due to ASIC limitations in identifying the egress packet type correctly.

Configuration Example: Explicit Null

This example shows how to configure explicit null label.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# address-family ipv4
RP/0/RP0/CPU0:Router(config-ldp-af)# label local advertise explict-null
RP/0/RP0/CPU0:Router(config-ldp-af)# commit
```

Configuration Example: Implicit Null Override

This example shows how to configure implicit null override for a set of prefixes.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# address-family ipv4
RP/0/RP0/CPU0:Router(config-ldp-af)# label local advertise implicit-null-override for acl-1
RP/0/RP0/CPU0:Router(config-ldp-af)# commit
```

Label Distribution Protocol Auto-configuration

LDP auto-configuration allows you to automatically configure LDP on all interfaces for which the IGP protocol is enabled. Typically, LDP assigns and advertises labels for IGP routes and must often be enabled on all active interfaces by an IGP. During LDP manual configuration, you must define the set of interfaces under LDP which is a time-intensive procedure. LDP auto-configuration eliminates the need to specify the same list of interfaces under LDP and simplifies the configuration tasks.

Configuration Example: Enabling LDP Auto-Configuration for OSPF

This example shows how to enable LDP auto-configuration for a specified OSPF instance.

```
RP/0/RP0/CPU0:Router(config)# router ospf 190
RP/0/RP0/CPU0:Router(config-ospf)# mpls ldp auto-config
RP/0/RP0/CPU0:Router(config-ospf)# area 8
RP/0/RP0/CPU0:Router(config-ospf-ar)# interface HundredGigE 0/0/0/5
RP/0/RP0/CPU0:Router(config-ospf-ar-if)# commit
```

Configuring Session Protection

When a new link or node comes up after a link failure, IP converges earlier and much faster than MPLS LDP and may result in MPLS traffic loss until the MPLS convergence. If a link flaps, the LDP session also flaps due to loss of link discovery. LDP session protection minimizes traffic loss, provides faster convergence, and protects existing LDP (link) sessions. When session protection is enabled for a peer, LDP starts sending targeted hello (directed discovery) in addition to basic discovery link hellos. When the direct link goes down, the targeted hellos can still be forwarded to the peer LSR over an alternative path as long as there is one. So, the LDP session stays up after the link goes down.

You can configure LDP session protection to automatically protect sessions with all or a given set of peers (as specified by peer-acl). When configured, LDP initiates backup targeted hellos automatically for neighbors for which primary link adjacencies already exist. These backup targeted hellos maintain LDP sessions when primary link adjacencies go down.

Configuration Example

This example shows how to configure LDP session protection for peers specified by the access control list peer-acl-1 for a maximum duration of 60 seconds.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# session protection for peer-acl-1 duration 60
RP/0/RP0/CPU0:Router(config-ldp)# commit
```

Configuring Label Distribution Protocol- Interior Gateway Protocol (IGP) Synchronization

Lack of synchronization between LDP and Interior Gateway Protocol (IGP) can cause MPLS traffic loss. Upon link up, for example, IGP can advertise and use a link before LDP convergence has occurred or, a link may continue to be used in IGP after an LDP session goes down.

LDP IGP synchronization coordinates LDP and IGP so that IGP advertises links with regular metrics only when MPLS LDP is converged on that link. LDP considers a link converged when at least one LDP session is up and running on the link for which LDP has sent its applicable label bindings and received at least one label binding from the peer. LDP communicates this information to IGP upon link up or session down events and IGP acts accordingly, depending on sync state.

LDP-IGP synchronization is supported for both OSPF and ISIS protocols and is configured under the corresponding IGP protocol configuration mode. Under certain circumstances, it might be required to delay declaration of re-synchronization to a configurable interval. LDP provides a configuration option to delay declaring synchronization up for up to 60 seconds. LDP communicates this information to IGP upon linkup or session down events.

From the 7.1.1 release, you can configure multiple MPLS-TE tunnel end points on an LER using the TLV 132 function in IS-IS. You can configure a maximum of 63 IPv4 addresses or 15 IPv6 addresses on an LER.

Configuring LDP IGP Synchronization: Open Shortest Path First (OSPF) Example

This example shows how to configure LDP-IGP synchronization for an OSPF instance. The synchronization delay is configured as 30 seconds.

```
RP/0/RP0/CPU0:Router(config)# router ospf 100
RP/0/RP0/CPU0:Router(config-ospf)# mpls ldp sync
RP/0/RP0/CPU0:Router(config-ospf)# mpls ldp igp sync delay 30
RP/0/RP0/CPU0:Router(config-ospf)# commit
```

Configuring LDP IGP Synchronization: Intermediate System to Intermediate System (IS-IS)

This example shows how to configure LDP-IGP synchronization for IS-IS.

```
RP/0/RP0/CPU0:Router(config)# router isis 100
RP/0/RP0/CPU0:Router(config-isis)#  interface HundredGigE 0/0/0/5
RP/0/RP0/CPU0:Router(config-isis-if)# address-family ipv4 unicast
RP/0/RP0/CPU0:Router(config-isis-if-af)# mpls ldp sync
RP/0/RP0/CPU0:Router(config-isis-if-af)# commit
```

Configuring Label Distribution Protocol Graceful Restart

LDP Graceful Restart provides a mechanism for LDP peers to preserve the MPLS forwarding state when the LDP session goes down. Without LDP Graceful Restart, when an established session fails, the corresponding forwarding states are cleaned immediately from the restart and peer nodes. In this case, LDP forwarding has to restart from the beginning, causing a potential loss of data and connectivity. If LDP graceful restart is configured, traffic can continue to be forwarded without interruption, even when the LDP session restarts. The LDP graceful restart capability is negotiated between two peers during session initialization time. During session initialization, a router advertises its ability to perform LDP graceful restart by sending the graceful restart typed length value (TLV). This TLV contains the reconnect time and recovery time. The values of the

reconnect and recovery times indicate the graceful restart capabilities supported by the router. The reconnect time is the amount of time the peer router waits for the restarting router to establish a connection. When a router discovers that a neighboring router is restarting, it waits until the end of the recovery time before attempting to reconnect. Recovery time is the amount of time that a neighboring router maintains its information about the restarting router.

Configuration Example

This example shows how to configure LDP graceful restart. In this example, the amount of time that a neighboring router maintains the forwarding state about the gracefully restarting router is specified as 180 seconds. The reconnect time is configured as 169 seconds.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)#  interface HundredGigE 0/0/0/5
RP/0/RP0/CPU0:Router(config-ldp-if)# exit
RP/0/RP0/CPU0:Router(config-ldp)# graceful-restart
RP/0/RP0/CPU0:Router(config-ldp)# graceful-restart forwarding-state-holdtime 180
RP/0/RP0/CPU0:Router(config-ldp)# graceful-restart reconnect-timeout 169
RP/0/RP0/CPU0:Router(config-ldp)# commit
```

Configuring Label Distribution Protocol Nonstop Routing

LDP nonstop routing (NSR) functionality makes failures, such as Route Processor (RP) or Distributed Route Processor (DRP) fail over, invisible to routing peers with minimal to no disruption of convergence performance. By default, NSR is globally enabled on all LDP sessions except AToM.

A disruption in service may include any of these events:

- Route processor (RP) or distributed route processor (DRP) failover
- LDP process restart
- Minimum disruption restart (MDR)

Note Unlike graceful restart functionality, LDP NSR does not require protocol extensions and does not force software upgrades on other routers in the network, nor does LDP NSR require peer routers to support NSR. L2VPN configuration is not supported on NSR. Process failures of active LDP results in session loss and, as a result, NSR cannot be provided unless RP switchover is configured as a recovery action.

Configuration Example

This example shows how to configure LDP Non-Stop Routing.

```
RP/0/RP0/CPU0:Router(config)# mpls ldp
RP/0/RP0/CPU0:Router(config-ldp)# nsr
RP/0/RP0/CPU0:Router(config-ldp)# commit
```

Verification

```
RP/0/RP0/CPU0:Router# show mpls ldp nsr summary
```

```
Mon Dec  7 04:02:16.259 UTC
Sessions:
Total: 1, NSR-eligible: 1, Sync-ed: 0
(1 Ready)
```

Configuring LDPv6

The LDP configuration model is extended to introduce IPv6 as an option under the address family submodes that reside under LDP global and interface configurations. IPv6 address family is available under LDP global, LDP VRF global and interface configurations. LDPv6 is supported only under default VRF. LDPv6 should be enabled on all interfaces that connects the router to potential LDPv6 peer routers.

Note In the 7.2.1 release, LDPv6 is only supported for routers in an LSR role, and not the LDP Label Edge Router (LER) role.

Restrictions for MPLS LDPv6

MPLS LDPv6 has the following restrictions:

- IPv6 address family is supported only under default VRF.
- Implicit enabling of IPv6 address family is not allowed. It needs explicit enabling.
- It is recommended to configure a routable IPv6 discovery transport address when only LDP IPv6 is configured without explicitly specifying a router-id.

IPv6 Support in MPLS LDP

MPLS LDPv6 makes the LDP control plane to run on IPv6 in order to setup LSPs for IPv6 prefixes. This support enables most of the LDP functions supported on IPv4 to be extended to IPv6. In this context, support for native MPLS LDP over IPv6 is provided in order to seamlessly continue providing existing services while enabling new ones.

LDP associates a forwarding equivalence class (FEC) with each label switched path (LSP) it creates. The FEC associated with an LSP specifies which packets are mapped to that LSP. LDP establishes sessions with peers and exchanges FEC label bindings with them to enable creation of LSPs to carry MPLS traffic destined to IP prefixes.

As per RFC 5036, LDP base specification defines procedures and messages for exchanging bindings for IPv4 and IPv6 addresses and routing prefixes. LDPv6 related RFCs explain control plane and binding advertisement support for LDPv6.

The procedures of address bindings, label bindings, and forwarding setup are same for IPv4 and IPv6 address families in LDP. The only difference is that a different address format is used according to the IP address family. While a single-stack IP address family (IPv4-only or IPv6-only) enabled interfaces between a set of routers is the most typical deployment, scenarios for LSR interconnections using both IPv4 and IPv6 interfaces are also supported.

IPv6 support in MPLS LDP implements draft-ietf-mpls-ldp-ipv6 version12.

LDPv6 Function

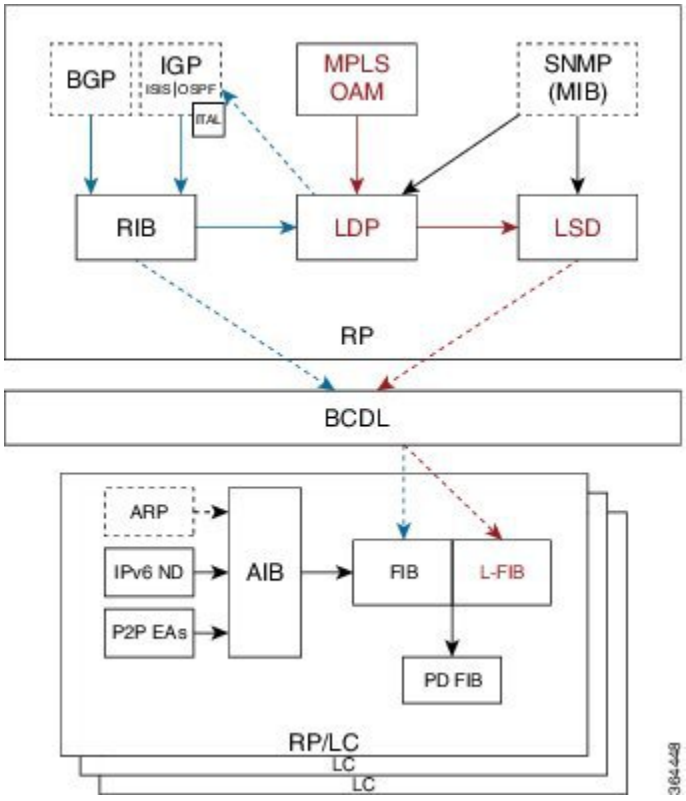
LDP functionality can be broadly divided into two categories, control plane and LSR setup.

- Control plane includes these functions - neighbor discovery (hello adjacencies), transport connection/endpoint (TCP connection), session and peering, and bindings exchange.
- LSP setup includes these functions - acquire FEC information through RIB, assign and advertise local label bindings for FEC, advertise local (interface) IP address bindings and setup forwarding rewrites.

For the control plane, the underlying address family can be either IPv4-only, IPv6-only or both. Whereas for the LSP setup, an LSP is setup for IPv4 or IPv6 FEC prefix.

This figure illustrates the main LDPv6 components.

Figure 1: LDP IPv6 Architecture

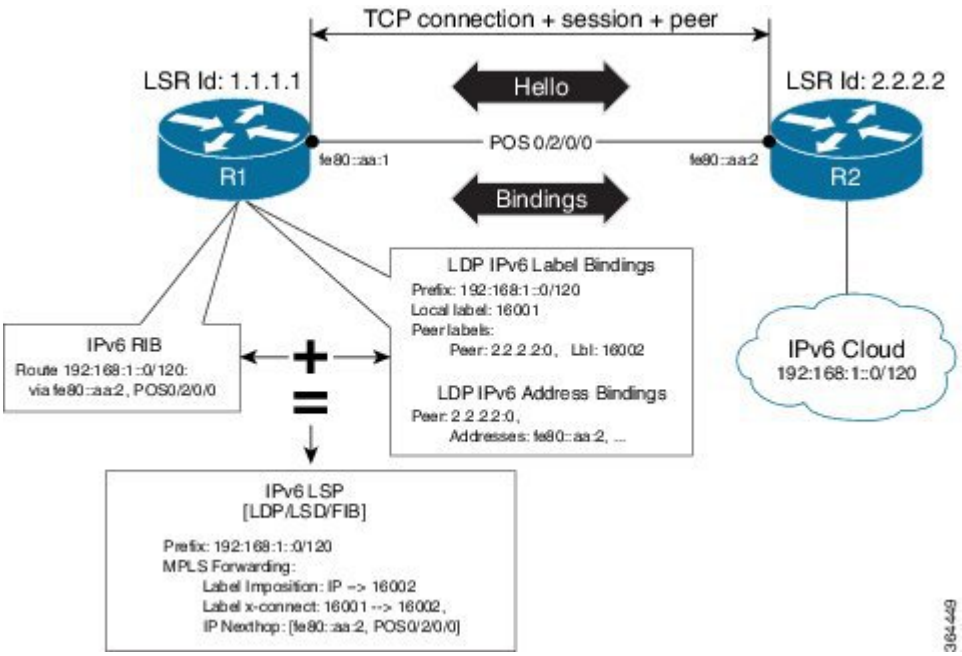


LDP functions in an MPLS LDPv6 setup:

- Receive routing updates from routing information base (RIB) for global IPv6 prefixes
- Assign local labels for IPv6 prefixes
- Receive IPv6 address or state notifications for local IPv6 enabled interfaces from IP Address Repository Manager (IP-ARM/IM) and LAS for IPv6 link-local unicast addresses
- Advertise/Accept IPv6 label bindings and address bindings to/from peers
- Setup MPLS forwarding to create IPv6 LSPs

- Provide IPv6 LSP information to MPLS OAM as and when requested
- Service MIB requests for IPv6 control plane queries and generate MIB traps
- Provide LDPv6 convergence status for a link to IGP for LDP-IGP Sync feature for IPv6

Figure 2: A high level depiction of LDPv6 Control Plane and LSP Setup



Topology Scenarios

A typical deployment scenario consists of single-stack IP address-family (IPv4-only or IPv6-only) enabled interfaces between a set of routers. The following are some topology scenarios, and a description of the control plane and LSP setup scenarios.

Here, R2 is the reference router.

One dual-stack interface/same neighbor

```
R1_ _ _ _ _ _ _ _R2

IPv4+IPv6
```

- Neighbor Discovery - IPv4 and IPv6 Hellos are sent on the interface to R1.
- Transport Connection - IPv4 endpoints or IPv6 endpoints (as per user preference).
- Label binding exchange - IPv4 and IPv6 prefixes.
- Address binding exchange - IPv4 and IPv6 addresses.
- LSPs - IPv4 and IPv6 over the same nexthop interface to R1.

Two single-stack interfaces/same neighbor:

```
1. (IPv4)

R1_ _ _ _ _ _ _ _ _R2
```

```
_ _ _ _ _ _ _ _ _
2. (IPv6)
```

- Neighbor Discovery - IPv4 Hellos on interface-1 to R1, and IPv6 Hellos on interface-2 to R1.
- Transport Connection - IPv4 endpoints or IPv6 endpoints (as per user preference).
- Label binding exchange - IPv4 and IPv6 prefixes.
- Address binding exchange - IPv4 and IPv6 addresses.
- LSPs - IPv4 over nexthop interface-1 to R1, and IPv6 over nexthop interface-2 to R1.

Two single-stack interfaces/different neighbors with different address families:

```
1. (IPv4) 2. (IPv6)

R1_ _ _ _ _ _ _ _ R2_ _ _ _ _ _ _ _ R3
```

- Neighbor Discovery - IPv4 Hellos on interface-1 to R1, and IPv6 Hellos on interface-2 to R3.
- Transport Connection - IPv4 endpoints with R1 and IPv6 endpoints with R3.
- Label binding exchange - IPv4 and IPv6 prefixes to R1 and R3.
- Even if all three LSRs are dual-stack, traffic from R1 to R3 will not be completely labeled.
- If there is IPv6 traffic, it is unlabeled from R1 to R2. Labels are imposed only at R2 (although in this specific case implicit null imposition) to R3.
- If there is IPv4 traffic, it is labeled from R1 to R2. But the traffic will go unlabeled between R2 and R3 given that no IPv4 adjacency exists between R2 and R3
- Address binding exchange - IPv4 and IPv6 addresses to R1 and R3
- LSPs - IPv4 over nexthop interface-1 to R1 and IPv6 over nexthop interface-2 to R3

Feature Support in LDPv6

The following features are supported in LDPv6:

- Single-stack (native IPv6) and dual-stack (IPv4+IPv6) topologies.
- New operating modes in LDP:
  - Native LDPv6
  - LDPv6 over IPv4 and LDPv4 over IPv6 connection endpoints

    LDP Hellos carry optional transport address type length value (TLV) to notify a peer about TCP or transport connection endpoint. An LSR can include either IPv4 or IPv6 transport address TLV in an IPv4 or IPv6 Hello message. There is no difference in the TLV format of transport address for IPv4 and IPv6.

    Only one transport connection is established between two discovered peers, whether there be single address family Hello adjacencies or multi-address family (both IPv4 and IPv6) Hello adjacencies.

    In a dual-stack setup, when LDP has the option to establish transport connection either using IPv4 endpoints or IPv6 endpoints, IPv6 connection is preferred over IPv4 connection. If LDP is locally

enabled for both IPv4 and IPv6 address families, every new session is treated as potential dual-stack connection. Under such circumstances, IPv6 preference is kept in place for maximum fifteen seconds for the session to establish, after which the LDP tries to establish a connection with the peer using IPv4. A user can override this default behavior by specifying the preference for a set of dual-stack peers to use IPv4 transport for the connection. Furthermore, a user may also specify maximum wait time to wait to establish the preferred transport connection. If the preferred transport establishment times out, LDP tries to establish connection with other non-preferred transport address families. This applies to both the cases when an LSR acts as active side or passive side for the TCP connection.

To override default IPv6 transport preference for dual-stack cases, use the **mpls ldp neighbor dual-stack transport-connection prefer ipv4** command. To specify the maximum time the preferred address family connection must wait to establish a connection before resorting to a non-preferred address family, use the **mpls ldp neighbor dual-stack transport-connection max-wait** command.

Once a transport connection is established, it is not torn down depending on preferences. If the address family related to established transport connection is disabled under LDP, the corresponding transport connection is reset to reestablish the connection.

For a single-stack setup, there is no contention; the transport connection uses the given address family.

- LDP Control Plane is IPv6 aware
- LDPv6 LSP forwarding setup - LDP interacts with LSD in order to setup IPv6 LSP forwarding. The steps involved in this interaction are:
  - Label allocation for an IPv6 prefix is learnt from RIB.
  - Setup imposition and label switching forwarding path for given IPv6 prefix by creating IPv6 forwarding rewrites.
  - Like LDPv4, rewrite delete and label free operations are performed when a route disappears or is disallowed under LDP due to label policy.
  - There is no new requirement related to MPLS enabling or disabling. LDP also MPLS-enables in LSD (if not already) any LDP enabled interface, which is in the *UP* state for IP4 and/or IPv6 and has IPv4 and/or IPv6 addresses assigned.
  - In case of dual-stack LDP, a single Resource-Complete is sent by LDP to LSD once RIB-Converged notification is received for both IPv4 and IPv6 redistribute tables.
- Distribution of IPv4 and IPv6 bindings over a single LDP session established over IPv4 or IPv6
- LDP Downstream on Demand
- LDP session protection

  LDP session protection is a feature to protect an LDPv6 session. In case of dual-stack hello adjacencies with a peer, there is only a single targeted hello adjacency to protect the session. Session protection forms targeted adjacency of address family same as the transport connection. For IPv6, the target of the session protection is the remote transport connection endpoint. For IPv4, the target of the session protection is remote LSR ID.
- LDP IGPv6 sync on IPv6 interface

  This feature lets IGP support LDP IGP Sync feature for IPv6 address family. This means that Intermediate System-to-Intermediate System (IS-IS) allows IGP under an interface's IPv6 address family, whereas

OSPFv3 implements it just like existing support in OSPF for IPv4. When the IGP Sync feature is enabled, LDP convergence status on an interface is considered by the IGP under the context of a given address family. This behavior applies to IGP Sync for both non-TE as well as TE tunnel interfaces.

- LDP Typed Wildcard for IPv6 prefix FEC

  This feature adds support for Typed Wildcard for IPv6 Prefix FEC. The support includes:

  - Being able to send or receive IPv6 Prefix Typed Wildcard FEC element in label messages.
  - Respond to Typed Wildcard Label Requests received from peer by replaying its label database for IPv6 prefixes.
  - Make use of Typed Wildcard Label Requests towards peers to request replay of peer label database for IPv6 prefixes. For example, on local inbound policy changes.

- Label allocation, advertisement and accept policies for IPv6 prefixes
- Local label assignment and advertisement for IPv6 default-route (::/0)
- Session MD5 authentication for IPv6 transport
- IPv6 Explicit-Null label

  IPv6 explicit null label feature support includes:

  - Advertisement and receipt of IPv6 explicit-null label to and from peers.
  - IPv6 explicit-null outgoing label in forwarding setup.
  - Explicit-null advertisement policy for a set of IPv6 prefixes and/or set of peers.
  - Explicit-null configuration change. Change in explicit-null configuration is handled by first transferring a wildcard withdraw with null label to peer(s), followed by advertising the appropriate null (implicit or explicit) label to the peer(s) again. This works without any issue as long as a single IP address family is enabled. In case of a dual-stack LSR peer, a change of configuration related to explicit-null advertisement for a given address family may cause unnecessary mix-up in the other address family.

- LDPv6 LFA FRR

  Local LFA FRR for IPv6 is supported. However, it is required that the primary and backup paths are of the same address family type, that is, an IPv4 primary path must not have an IPv6 backup path.

- NSF for LDPv6 traffic

  Non-stop forwarding (NSF) support is either provided through LDP NSR or graceful restart mechanisms.

- IGP/LDP NSR for IPv6
- IGP/LDP Graceful Restart for IPv6
- MPLS OAM: New FECs

  LSPV supports two new FECs.

  - LDPv6 Prefix FEC Encoding/Decoding

Label Switched Path Verification (LSPV) encodes/decodes the LDP IPv6 Prefix FEC. Prefix is in the network byte order and the trailing bits are to be set to zero when prefix length is shorter than 128 bits.

- Generic IPv6 Prefix FEC Encoding/Decoding

  LSPV encodes/decodes the generic IPv6 Prefix FEC. Prefix is in the network byte order and the trailing bits are to be set to zero when prefix length is shorter than 128 bits.

  Generic IPv6 FEC is used in addition to the LDPv6 FEC. This serves the following primary purposes:

  - Allows user to perform LSP ping and traceroute to verify data plane without involving control plane of the FEC in echo request and response.
  - If support for a new FEC is preferred in the future, the generic FEC can be used until corresponding control plane is explicitly supported by LSPV.

- IPv6 LSR MIB

  MPLS OAM LDP MIBS is extended to support IPv6. All LSR MIB objects that reference an InSegment prefix and OutSegment next hop address are modified to support IPv6.

- LSP ping support for LDPv6
- LSP trace-route support for LDPv6
- LSP tree-trace support for LDPv6

Scale

- The same support as LDPv4 native is provided for LDPv6 native scale
- Dual-stack–The aggregate scale of LDPv4 and LDPv6 is the same as the currently support for LDPv4 native scale

Unsupported Features in LDPv6

- LDPv6 over TEv4 (traffic engineering)
- Interfaces
  - LDP auto-config for IPv6
  - LDPv6 over TEv6
  - LDPv6 over GREv6
- LDP auto-config for IPv6 IGP
- LDP Label Edge Router (LER) function
- Remote LFA FRR
- L2VPN
  - L2VPN over IPv6 LSPs
  - L2VPN signaling with LDP when the nexthop address is IPv6

- IPv6 BGP Redistribution
- Applications with native LDPv6
  - Multicast extension to LDP (mLDP) for IPv6 FEC with label binding through IPv4 and IPv6 transport
  - ICCP - ICCP and LDP ICCP with IPv6 neighbor node
  - PW
- L3VPN
  - Native IPv6 MPLS L3VPNs
  - 4PE
  - 4vPE
  - LDPv6 CSC

IPv6 Label Bindings

LDP stores label bindings associated with FEC prefix in its Label Information Base (LIB) [TIB in Cisco LDP]. An entry in LIB corresponds to a prefix and holds the following bindings:

- Local binding: Local label assigned for this prefix (which is learnt through local RIB)
- Remote bindings: Array of peer labels (prefix-label bindings received in label mapping message from peers)

An entry in LIB can exist due to local binding presence, or due to remote binding(s) presence, or due to both local and remote bindings presence. The forwarding setup, however, mandates that local binding be present for a prefix.

Extensions have been implemented to support IPv6 prefixes for LIB in LDP. For per-address family convergence or preference reasons, separate or new LIB is implemented to keep and maintain IPv6 prefixes. In case of dual-stack LDP, LIBv4 is preferred over LIBv6 wherever possible. For example, during background *housekeeping* function, LIBv4 is processed before LIBv6.

IPv6 Address Bindings

LDP needs to maintain IPv6 address database for local and peer interface addresses. The IPv4 address module for local/peer addresses is extended to keep IPv4/IPv6 addresses in their respective databases, much like LIB database. In case of a dual-stack LDP, IPv4 local address database function is preferred over IPv6 local address database function where ever possible.

LDP Control Plane: Bindings Advertisement

LDP base specification allows exchange of IPv4/IPv6 bindings (address/label) on an established session. When both IPv4 and IPv6 address families are enabled under LDP, LDP distributes address/label bindings for both address families to its established peer according to local policies. Following are a few significant points pertaining to bindings support for IPv6:

- LDP allocates/advertises local label bindings for link-local IPv6 address prefixes. If received, such FEC bindings are ignored

- LDP sends only the Prefix FEC of the single address family type in a FEC TLV and not include both. If such a FEC binding is received, the entire message is ignored
- LDP sends only the addresses belonging to same address family in a single address list TLV (in address or address withdraw message)

If an address family is not enabled on receiving LSR, LDP discards any bindings received from peer(s) for the address family. This means that when address family is enabled, LDP needs to reset existing sessions with the peers in order to re-learn the discarded bindings. The implementation is optimized to reset only those sessions which were previously known to be dual-stack and had sent bindings for both address families.

LSP Mapping

LDP uses IPv6 adjacency information instead of IP address to map an IPv6 link-local nexthop to an LDP peer.

In addition to other usual checks before using a label from nexthop LDP peer, LDP uses the nexthop label for a prefix of a given address family, if there are one or more LDP hello adjacencies of the same address family type established with the peer.

Label Policies

LDP allows a user to configure label policies for allocation, acceptance, receipt, and advertisement of labels for the given prefixes.

Following are the significant points pertaining to the IPv6 support for label policies:

- Label policies and their configurations are allowed under address family IPv6
- Any policy that specifies prefix or a set of prefixes through an ACL, supports both IPv4 and IPv6 variants for address(s) or ACLs
- Any policy that specifies peer address or set of peer addresses through an ACL, supports both IPv4 and IPv6 variant for peer address(s) or ACL
- Any policy that specifies the peer's LSR ID in a peer ACL continues to take IPv4 ACL based policy irrespective of the feature configuration

Dual-Stack Capability TLV

Clear rules are specified in RFC 5036 to determine transport connection roles in setting up a TCP connection for single-stack LDP. But RFC 5036 is not clear about dual-stack LDP, in which an LSR may assume different roles for different address families, causing issues in establishing LDP sessions.

To ensure a deterministic transport connection role for the dual-stack LDP, the dual-stack LSR conveys its transport connection preference in every LDP Hello message. This preference is encoded in a new TLV (Type Length Value) called the Dual-Stack Capability TLV. Dual-stack LSR always checks for the presence of the dual-stack capability TLV in the received LDP Hello messages and takes appropriate action for establishing or maintaining sessions.

RFC 7552 specifies more details about updates to LDPv6.

Dual-Stack Capability TLV Format

```
0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0|  Dual-Stack Capability   |            Length             |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TR |     Reserved          |            MBZ                |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Dual-Stack Capability TLV Fields

Field	Description
U and F bits	1 and 0 (as specified by RFC 5036)
Dual-Stack Capability	TLV code point (0x0701)
TR: Transport Connection Preference	TR: Transport Connection Preference: • 0100: LDPoIPv4 connection • 0110: LDPoIPv6 connection (default)
Reserved	This field is reserved. It must be set to zero on transmission and ig receipt
MBZ	Must be zero

Compliance Check

The compliance check prevents sessions being formed with prior RFC 7552 implementation of LDPv6.

If the dual-stack capability TLV is not present in the received Hellos and the compliance check is configured, the local and remote preferences must match to establish a session. If the preferences do not match, the LDP Hellos are dropped and the session is not established. Compliance check has therefore been disabled by default.

Use the **neighbor dual-stack tlv-compliance** command in MPLS LDP configuration to enable the compliance check.

Configuring ISIS for IPv6 and LDPv6

Intermediate System-to-Intermediate System (IS-IS) is an Interior Gateway Protocol (IGP) that advertises link-state information throughout the network to create a picture of the network topology. IPv6 IS-IS extends the address families supported by IS-IS to include IPv6, in addition to IPv4.

Previously, IS-IS supported registration of only LDP IPv4 sync status change. This has now been enhanced to support registration of notifications of LDP IPv6 sync status change. IS-IS determines the link-metrics to be advertised based on the LDP-IGP sync status on the IPv4 and IPv6 address families.

IS-IS supports non-stop forwarding (NSF) by preserving the LDPv6-IGP sync status across high availability (HA) events of IS-IS process restarts and failover.

IS-IS also supports LDPv6-IGP sync for LFA-FRR by checking the sync status of the backup interface (if it is configured with LDP IPv6 sync).

Configure Interfaces With IPv6 Addresses

```
Router# configure
Router(config)# interface Loopback 0
Router(config-if)# ipv6 address 6:6:6::6/128

Router(config)# interface HundredGigE0/0/0/0
Router(config-if)# ipv6 address 16:1::6/120
```

Enable IS-IS

Enable ISIS routing protocol and associate the IPv6 interfaces to it. Enable IPv6 capability within ISIS.

```
Router(config)# router isis 100
Router(config-isis)# net 49.0000.0000.0000.0006.00
Router(config-isis)# interface Loopback 0
Router(config-isis-if)# address-family ipv6 unicast

Router(config-isis)# interface HundredGigE0/0/0/0
Router(config-isis-if)# address-family ipv6 unicast
```

Implement LDPv6 Globally

```
Router(config)# mpls ldp
Router(config-ldp)# address-family ipv6
```

Configure Default Transport Address

LDP computes default local transport address for IPv6 from its IPv6 interface or address database by picking the lowest operational loopback interface with global unicast IPv6 address. This means that any change in this loopback state or address, flaps or changes the default transport address for IPv6 and may cause session flaps using such an address as transport endpoint. For example, if a session is currently active on Loopback2 as during it's inception it was the lowest loopback with an IPv6 address, and a lower loopback, Loopback0, is configured with an IPv6 address, the session does not flap. However, if it does flap, the next time the session is attempted, Loopback0 is used.

The session flaps when configuring discovery transport address explicitly.

Use the **discovery transport-address** command under the LDP address family submode to specify the global transport address for IPv4 or IPv6.

Enable a global transport-address for the IPv6 address family.

It is recommended to configure global transport-address for IPv6 address family to avoid a potentially unstable default transport address.

```
Router(config-ldp-af)# discovery transport-address 5:6::78
```

Implement LDPv6 For The Physical Interface

```
Router(config-ldp)# interface HundredGigE0/0/0/0
Router(config-ldp-if)# address-family ipv6
```

Disable Implicit IPv4

The LDP configuration model was changed with the introduction of explicit address family enabling under LDP (VRF) global and LDP (VRF) interfaces. However, in order to support backward compatibility, the old configuration model was still supported for default VRF. There was, however, no option to disable the implicitly enabled IPv4 address family under default VRF's global or interface level.

A new configuration **mpls ldp default-vrf implicit-ipv4 disable** is now available to the user to disable the implicitly enabled IPv4 address family for the default VRF. The new configuration provides a step towards migration to new configuration model for the default VRF that mandates enabling address family explicitly. This means that if the new option is configured, the user has to explicitly enable IPv4 address family for default VRF global and interface levels. It is recommended to migrate to this explicitly enabled IPv4 configuration model.

IPv4 is implicitly enabled under default VRF and any LDP interface under default VRF. To operate as an IPv6-only LSR, disable the IPv4 address family.

```
Router(config-ldp)# default-vrf implicit-ipv4 disable
Router(config-ldp)# router-id 5.5.5.5
```

(Optional) Transport Preference Parameters

Configure IPv4 as the preferred transport (overriding the default setting of IPv6 as preferred transport) to establish connection for a set of dual-stack peers. You can also configure the maximum time (*max-wait*, in seconds) the preferred address family connection must wait to establish the transport connection before resorting to the non-preferred address family.

```
Router(config-ldp)# neighbor dual-stack transport-connection prefer ipv4
Router(config-ldp)# neighbor dual-stack transport-connection max-wait 5
```

LDPv6 Configuration Examples

The following example shows how to enable LDP IPv6 native under LDP. The user must enable IPv6 address family under LDP submodes.

```
configure
  mpls ldp
   address-family ipv6
   !
  !
```

The following example shows how to enable LDP IPv6 control plane on an LDP interface:

```
configure
  mpls ldp
   interface HundredGigE0/0/0/0
    address-family ipv6
    !
  !
```

The following examples shows how to configure IPv6-only LSR.

IPv4 is implicitly enabled under default VRF and any LDP interfaces under default VRF. In order to operate as an IPv6-only LSR, the user must also explicitly disable IPv4 address family.

LDPv6 Configuration Without Explicit IPv6 Export Address

In this example, there is no explicit IPv6 export address. The loopback's IPv6 address is used as the export address (6:6:6::6/128).

The router ID configured in MPLS LDP is not used in anyway for export. It is used only for LDP LSR identification.

```
configure
interface Loopback0
 ipv6 address 6:6:6::6/128
!
interface GigabitEthernet0/0/0/0
```

```
 ipv6 address 16:1::6/120
!
router isis 100
 net 49.0000.0000.0000.0006.00
 interface Loopback0
  address-family ipv6 unicast
  !
 !
 interface GigabitEthernet0/0/0/0
  address-family ipv6 unicast
  !
 !
mpls ldp
 default-vrf implicit-ipv4 disable
 router-id 6.6.6.6
 address-family ipv6
 !
 interface GigabitEthernet0/0/0/0
  address-family ipv6
 !
!
```

LDPv6 Configuration With Explicit IPv6 Export Address

In this example, there is an explicit IPv6 export address. However, there is no IPv6 loopback. There is no router-id configured, but the loopback IPv4 address is used.

```
Configure
!
interface Loopback0
 ipv6 address 6:6:6::6/128
!
interface HundredGigE0/0/0/0
 ipv6 address 16:1::6/120
!
router isis 100
 net 49.0000.0000.0000.0006.00
 address-family ipv6 unicast
 !
 interface Loopback0
  address-family ipv6 unicast
  !
 !
 interface HundredGigE0/0/0/0
  address-family ipv6 unicast
  !
 !
!
mpls ldp
 default-vrf implicit-ipv4 disable
 router-id 5.5.5.5
 neighbor
  dual-stack transport-connection max-wait 5
  dual-stack transport-connection prefer ipv4
 !
 address-family ipv6
  discovery transport-address 5:6::78
 !
 interface HundredGigE0/0/0/0
  address-family ipv6
 !
!
```

Associated Commands

- address-family (IS-IS)
- default-vrf implicit-ipv4 disable
- interface (IS-IS)
- ipv6 address
- mpls ldp
- neighbor dual-stack transport-connection prefer ipv4 for-peers

MPLS Label Distribution Protocol : Details

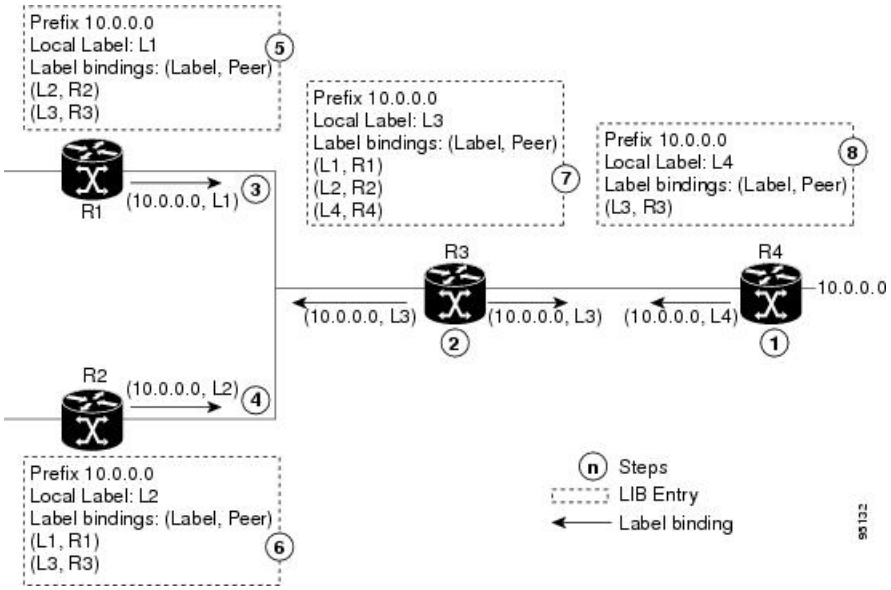
This section provides detailed conceptual information about setting up LSPs, LDP graceful restart, and LDP session protection.

Setting Up Label Switched Paths

MPLS packets are forwarded between the nodes on the MPLS network using Label Switched Paths(LSPs). LSPs can be created statically or by using a label distribution protocol like LDP. Label Switched Paths created by LDP performs hop-by-hop path setup instead of an end-to-end path. LDP enables label switched routers (LSRs) to discover their potential peer routers and to establish LDP sessions with those peers to exchange label binding information.

The following figure illustrates the process of label binding exchange for setting up LSPs.

Figure 3: Setting Up Label Switched Paths



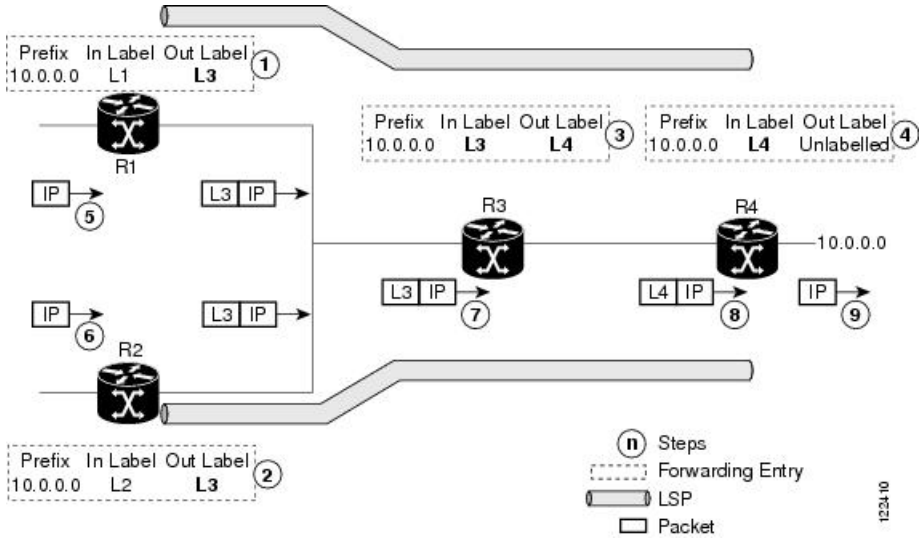
For a given network (10.0.0.0), hop-by-hop LSPs are set up between each of the adjacent routers (or, nodes) and each node allocates a local label and passes it to its neighbor as a binding:

1. R4 allocates local label L4 for prefix 10.0.0.0 and advertises it to its neighbors (R3).
2. R3 allocates local label L3 for prefix 10.0.0.0 and advertises it to its neighbors (R1, R2, R4).
3. R1 allocates local label L1 for prefix 10.0.0.0 and advertises it to its neighbors (R2, R3).
4. R2 allocates local label L2 for prefix 10.0.0.0 and advertises it to its neighbors (R1, R3).
5. R1's label information base (LIB) keeps local and remote labels bindings from its neighbors.
6. R2's LIB keeps local and remote labels bindings from its neighbors.
7. R3's LIB keeps local and remote labels bindings from its neighbors.
8. R4's LIB keeps local and remote labels bindings from its neighbors.

MPLS Forwarding

Once the label bindings are learned, MPLS forwarding plane is setup and packets are forwarded as shown in the following figure.

Figure 4: MPLS Forwarding



1. Because R3 is next hop for 10.0.0.0 as notified by the FIB, R1 selects label binding from R3 and installs forwarding entry (Layer 1, Layer 3).
2. Because R3 is next hop for 10.0.0.0 (as notified by FIB), R2 selects label binding from R3 and installs forwarding entry (Layer 2, Layer 3).
3. Because R4 is next hop for 10.0.0.0 (as notified by FIB), R3 selects label binding from R4 and installs forwarding entry (Layer 3, Layer 4).
4. Because next hop for 10.0.0.0 (as notified by FIB) is beyond R4, R4 uses NO-LABEL as the outbound and installs the forwarding entry (Layer 4); the outbound packet is forwarded IP-only.
5. Incoming IP traffic on ingress LSR R1 gets label-imposed and is forwarded as an MPLS packet with label L3.

6. Incoming IP traffic on ingress LSR R2 gets label-imposed and is forwarded as an MPLS packet with label L3.
7. R3 receives an MPLS packet with label L3, looks up in the MPLS label forwarding table and switches this packet as an MPLS packet with label L4.
8. R4 receives an MPLS packet with label L4, looks up in the MPLS label forwarding table and finds that it should be Unlabeled, pops the top label, and passes it to the IP forwarding plane.
9. IP forwarding takes over and forwards the packet onward.

Details of Label Distribution Protocol Graceful Restart

LDP (Label Distribution Protocol) graceful restart provides a control plane mechanism to ensure high availability and allows detection and recovery from failure conditions while preserving Nonstop Forwarding (NSF) services. Graceful restart is a way to recover from signaling and control plane failures without impacting forwarding.

Without LDP graceful restart, when an established session fails, the corresponding forwarding states are cleaned immediately from the restarting and peer nodes. In this case LDP forwarding restarts from the beginning, causing a potential loss of data and connectivity.

The LDP graceful restart capability is negotiated between two peers during session initialization time, in FT SESSION TLV. In this typed length value (TLV), each peer advertises the following information to its peers:

Reconnect time

Advertises the maximum time that other peer will wait for this LSR to reconnect after control channel failure.

Recovery time

Advertises the maximum time that the other peer has on its side to reinstate or refresh its states with this LSR. This time is used only during session reestablishment after earlier session failure.

FT flag

Specifies whether a restart could restore the preserved (local) node state for this flag.

Once the graceful restart session parameters are conveyed and the session is up and running, graceful restart procedures are activated.

When configuring the LDP graceful restart process in a network with multiple links, targeted LDP hello adjacencies with the same neighbor, or both, make sure that graceful restart is activated on the session before any hello adjacency times out in case of neighbor control plane failures. One way of achieving this is by configuring a lower session hold time between neighbors such that session timeout occurs before hello adjacency timeout. It is recommended to set LDP session hold time using the following formula:

```
Session Holdtime <= (Hello holdtime - Hello interval) * 3
```

This means that for default values of 15 seconds and 5 seconds for link Hello holdtime and interval respectively, session hold time should be set to 30 seconds at most.

Phases in Graceful Restart

The graceful restart mechanism is divided into different phases:

Control communication failure detection

Control communication failure is detected when the system detects either:

- Missed LDP hello discovery messages
- Missed LDP keepalive protocol messages
- Detection of Transmission Control Protocol (TCP) disconnection a with a peer

Forwarding state maintenance during failure

Persistent forwarding states at each LSR are achieved through persistent storage (checkpoint) by the LDP control plane. While the control plane is in the process of recovering, the forwarding plane keeps the forwarding states, but marks them as stale. Similarly, the peer control plane also keeps (and marks as stale) the installed forwarding rewrites associated with the node that is restarting. The combination of local node forwarding and remote node forwarding plane states ensures NSF and no disruption in the traffic.

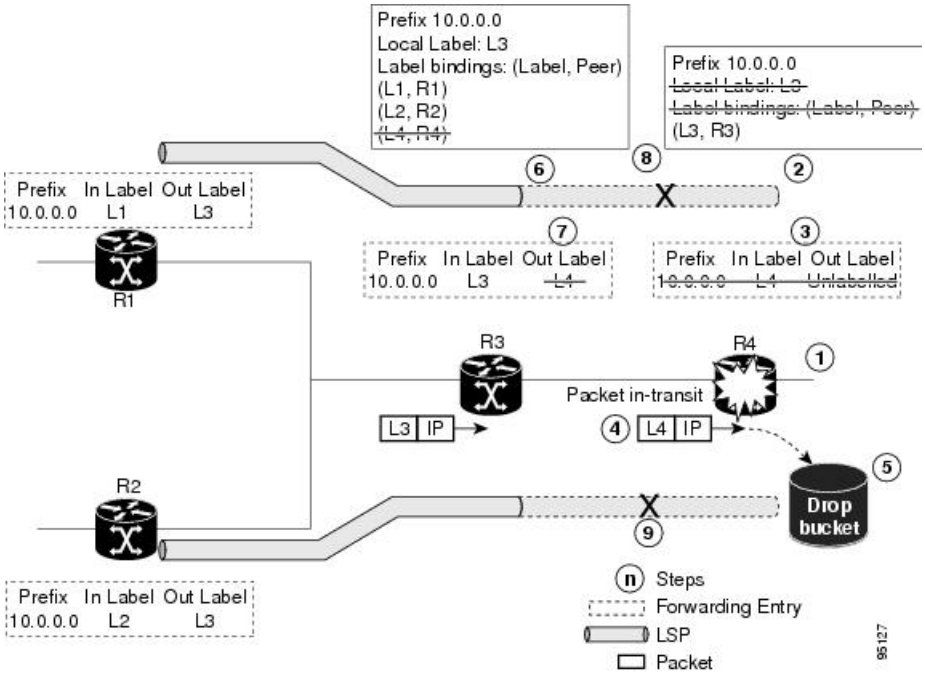
Control state recovery

Recovery occurs when the session is reestablished and label bindings are exchanged again. This process allows the peer nodes to synchronize and to refresh stale forwarding states.

Control Plane Failure

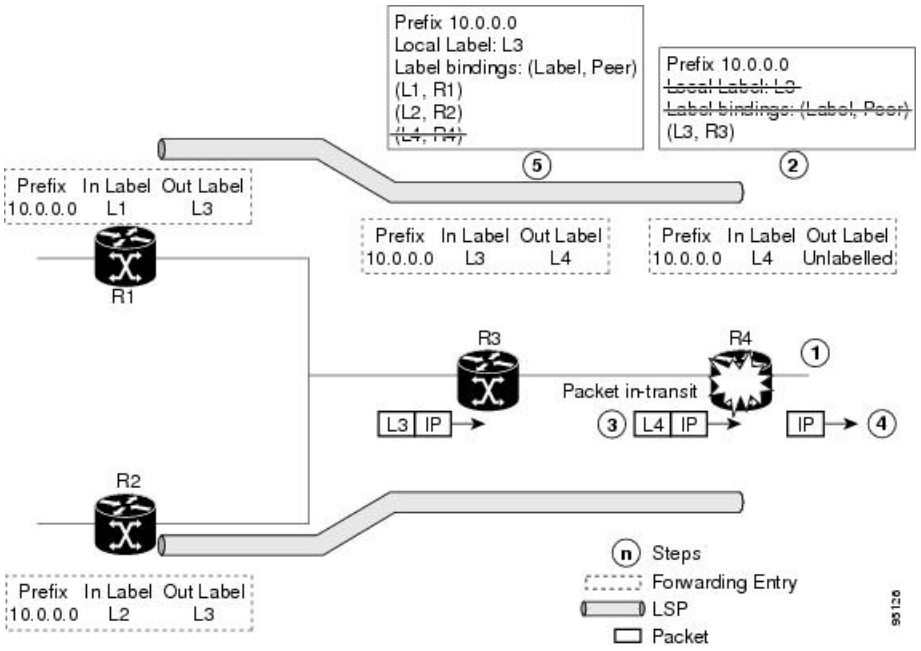
When a control plane failure occurs, connectivity can be affected. The forwarding states installed by the router control planes are lost, and the in-transit packets could be dropped, thus breaking NSF. The following figure illustrates control plane failure and recovery with graceful restart and shows the process and results of a control plane failure leading to loss of connectivity and recovery using graceful restart.

Figure 5: Control Plane Failure



Recovery with Graceful Restart

Figure 6: Recovering with Graceful Restart



1. The R4 LSR control plane restarts.
2. LIB is lost when the control plane restarts.
3. The forwarding states installed by the R4 LDP control plane are immediately deleted.
4. Any in-transit packets flowing from R3 to R4 (still labeled with L4) arrive at R4.
5. The MPLS forwarding plane at R4 performs a lookup on local label L4 which fails. Because of this failure, the packet is dropped and NSF is not met.
6. The R3 LDP peer detects the failure of the control plane channel and deletes its label bindings from R4.
7. The R3 control plane stops using outgoing labels from R4 and deletes the corresponding forwarding state (rewrites), which in turn causes forwarding disruption.
8. The established LSPs connected to R4 are terminated at R3, resulting in broken end-to-end LSPs from R1 to R4.
9. The established LSPs connected to R4 are terminated at R3, resulting in broken LSPs end-to-end from R2 to R4.

When the LDP control plane recovers, the restarting LSR starts its forwarding state hold timer and restores its forwarding state from the checkpointed data. This action reinstates the forwarding state and entries and marks them as old.

The restarting LSR reconnects to its peer, indicated in the FT Session TLV, that it either was or was not able to restore its state successfully. If it was able to restore the state, the bindings are resynchronized.

The peer LSR stops the neighbor reconnect timer (started by the restarting LSR), when the restarting peer connects and starts the neighbor recovery timer. The peer LSR checks the FT Session TLV if the restarting

peer was able to restore its state successfully. It reinstates the corresponding forwarding state entries and receives binding from the restarting peer. When the recovery timer expires, any forwarding state that is still marked as stale is deleted.

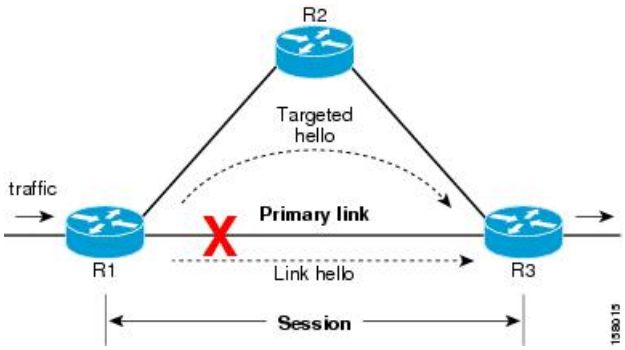
If the restarting LSR fails to recover (restart), the restarting LSR forwarding state and entries will eventually timeout and is deleted, while neighbor-related forwarding states or entries are removed by the Peer LSR on expiration of the reconnect or recovery timers.

Details of Session Protection

LDP session protection lets you configure LDP to automatically protect sessions with all or a given set of peers (as specified by peer-acl). When configured, LDP initiates backup targeted hellos automatically for neighbors for which primary link adjacencies already exist. These backup targeted hellos maintain LDP sessions when primary link adjacencies go down.

The Session Protection figure illustrates LDP session protection between neighbors R1 and R3. The primary link adjacency between R1 and R3 is directly connected link and the backup; targeted adjacency is maintained between R1 and R3. If the direct link fails, LDP link adjacency is destroyed, but the session is kept up and running using targeted hello adjacency (through R2). When the direct link comes back up, there is no change in the LDP session state and LDP can converge quickly and begin forwarding MPLS traffic.

Figure 7: Session Protection



Note When LDP session protection is activated (upon link failure), protection is maintained for an unlimited period time.

Controlling State Advertisements In An mLDP-Only Setup

This function explains the controlling of state advertisements of non-negotiated Label Distribution Protocol (LDP) applications. This implementation is in conformance with RFC 7473 (Controlling State Advertisements of Non-negotiated LDP Applications).

The main purpose of documenting this function is to use it in a Multipoint LDP (mLDP)-only environment, wherein participating routers don’t need to exchange any unicast binding information.

Non-Negotiated LDP Applications

The LDP capabilities framework enables LDP applications' capabilities exchange and negotiation, thereby enabling LSRs to send necessary LDP state. However, for the applications that existed prior to the definition of the framework (called *non-negotiated* LDP applications), there is no capability negotiation done. When an LDP session comes up, an LDP speaker may unnecessarily advertise its local state (without waiting for any capabilities exchange and negotiation). In other words, even when the peer session is established for Multipoint LDP (mLDP), the LSR advertises the state for these early LDP applications.

One example is *IPv4/IPv6 Prefix LSPs Setup* (used to set up Label Switched Paths [LSPs] for IP prefixes). Another example is *L2VPN P2P FEC 128 and FEC 129 PWs Signaling* (an LDP application that signals point-to-point [P2P] Pseudowires [PWs] for Layer 2 Virtual Private Networks [L2VPNs]).

In an mLDP-only setup, you can disable these non-negotiated LDP applications and avoid unnecessary LDP state advertisement. An LDP speaker that only runs mLDP announces to its peer(s) its disinterest (or non-support) in non-negotiated LDP applications. That is, it announces to its peers its disinterest to set up IP Prefix LSPs or to signal L2VPN P2P PW, at the time of session establishment.

Upon receipt of such a capability, the receiving LDP speaker, if supporting the capability, disables the advertisement of the state related to the application towards the sender of the capability. This new capability can also be sent later in a Capability message, either to disable a previously enabled application's state advertisement, or to enable a previously disabled application's state advertisement.

As a result, the flow of LDP state information in an mLDP-only setup is faster. When routers come up after a network event, the network convergence time is fast too.

IP Address Bindings In An mLDP Setup

An LSR typically uses peer IP address(es) to map an IP routing next hop to an LDP peer in order to implement its control plane procedures. mLDP uses a peer's IP address(es) to determine its upstream LSR to reach the root node, and to select the forwarding interface towards its downstream LSR. Hence, in an mLDP-only network, while it is desirable to disable advertisement of label bindings for IP (unicast) prefixes, disabling advertisement of IP address bindings will break mLDP functionality.

Uninteresting State - For the *Prefix-LSP* LDP application, *uninteresting* state refers to any state related to IP Prefix FEC, such as FEC label bindings and LDP Status. IP address bindings are not considered as an *uninteresting* state.

For the P2P-PW application LDP application, *uninteresting* state refers to any state related to P2P PW FEC 128 or FEC 129, such as FEC label bindings, MAC address withdrawal, and LDP PW status.

Control State Advertisement

To control advertisement of *uninteresting* state of non-negotiated LDP applications, the capability parameter TLV *State Advertisement Control Capability* is used. This TLV is only present in the Initialization and Capability messages, and the TLV can hold one or more State Advertisement Control (SAC) Elements.

As an example, consider two LSRs, S (LDP speaker) and P (LDP peer), that support all non-negotiated applications. S is participating (or set to participate) in an mLDP-only setup. Pointers for this scenario:

- By default, the LSRs will advertise state for all LDP applications to their peers, as soon as an LDP session is established.
- The **capabilities sac mldp-only** function is enabled on S.
- P receives an update from S via a Capability message that specifies to disable all four non-negotiated applications states.

- P's outbound policy towards S blocks and disables state for the unneeded applications.
- S only receives mLDP advertisements from specific mLDP-participating peers.

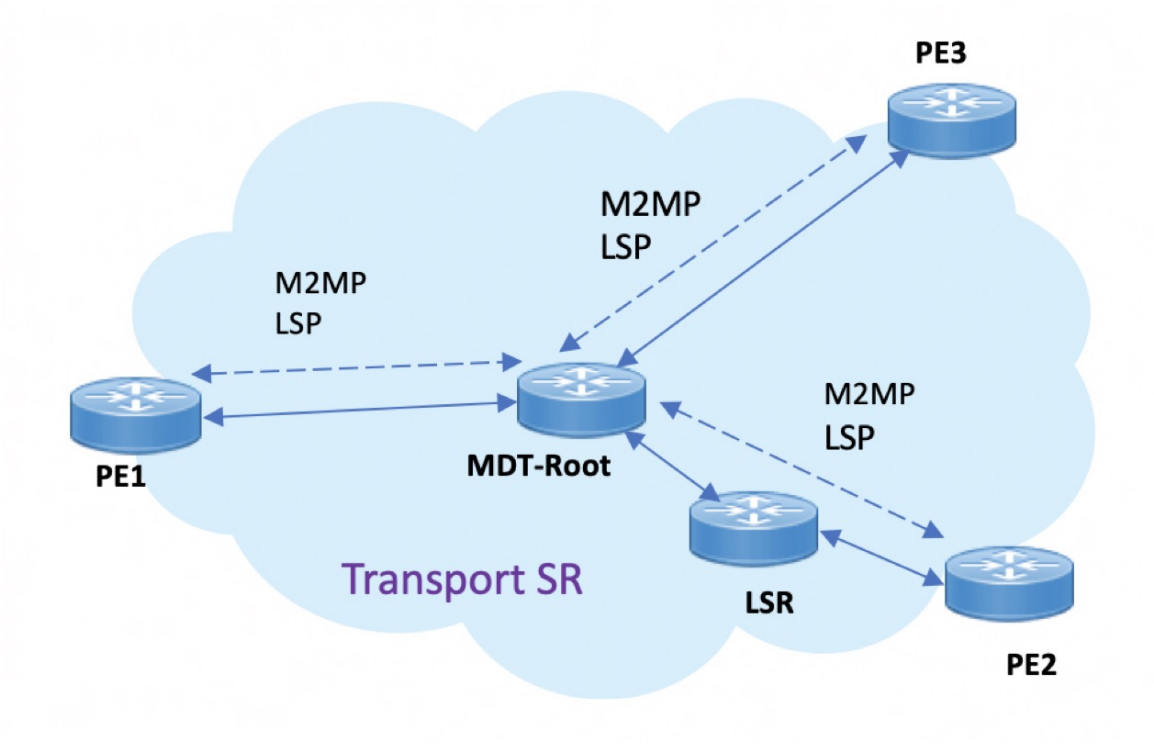
Use Cases For Controlling State Advertisements

Two use cases are explained, **mLDP-Based MVPN** and **Disable Prefix-LSPs On An L2VPN/PW tLDP Session**.

mLDP-Based MVPN

A sample topology and relevant configurations are noted below.

Figure 8: mLDP-Based MVPN Over Segment Routing



- The topology represents an MVPN profile 1 where an mLDP-based MVPN service is deployed over a Segment Routing core setup
- mLDP is required to signal MP2MP LSPs, whereas SR handles the transport.
- SAC capabilities are used to signal *mLDP-only* capability, which blocks unrequired unicast IPv4, IPv6, FEC128, and FEC129 related label binding advertisements.
- The **mldp-only** option is enabled on PE routers and P routers to remove unwanted advertisements.

Configuration

PE1 Configuration

Configure mLDP SAC capability on PE1.

```
PE1(config)# mpls ldp
PE1(config-ldp)# capabilities sac mldp-only
PE1(config-ldp)# commit
```

PE2 Configuration

Configure mLDP SAC capability on PE2.

```
PE2(config)# mpls ldp
PE2(config-ldp)# capabilities sac mldp-only
PE2(config-ldp)# commit
```

Verification

LDP peers (PE1 and PE2) are configured with **mldp-only** option, disabling all other SAC capabilities.

```
PE1# show running-config mpls ldp

mpls ldp
   capabilities sac mldp-only
   mldp
        address-family ipv4
        !

PE2# show running-config mpls ldp

mpls ldp
   capabilities sac mldp-only
   mldp
        address-family ipv4
        !
```

On PE1, verify PE2's SAC capabilities:

```
PE1# show mpls ldp neighbor 209.165.201.20 capabilities detail

Peer LDP Identifier: 209.165.201.20:0
  Capabilities:
    Sent:
      0x508  (MP: Point-to-Multipoint (P2MP))
      0x509  (MP: Multipoint-to-Multipoint (MP2MP))
      0x50b  (Typed Wildcard FEC)
      0x50d  (State Advertisement Control)
        [ {IPv4-disable}{IPv6-disable}{FEC128-disable}{FEC129-disable} ](length 4)
    Received:
      0x508  (MP: Point-to-Multipoint (P2MP))
      0x509  (MP: Multipoint-to-Multipoint (MP2MP))
      0x50b  (Typed Wildcard FEC)
      0x50d  (State Advertisement Control)
        [ {IPv4-disable}{IPv6-disable}{FEC128-disable}{FEC129-disable} ](length 4)
```

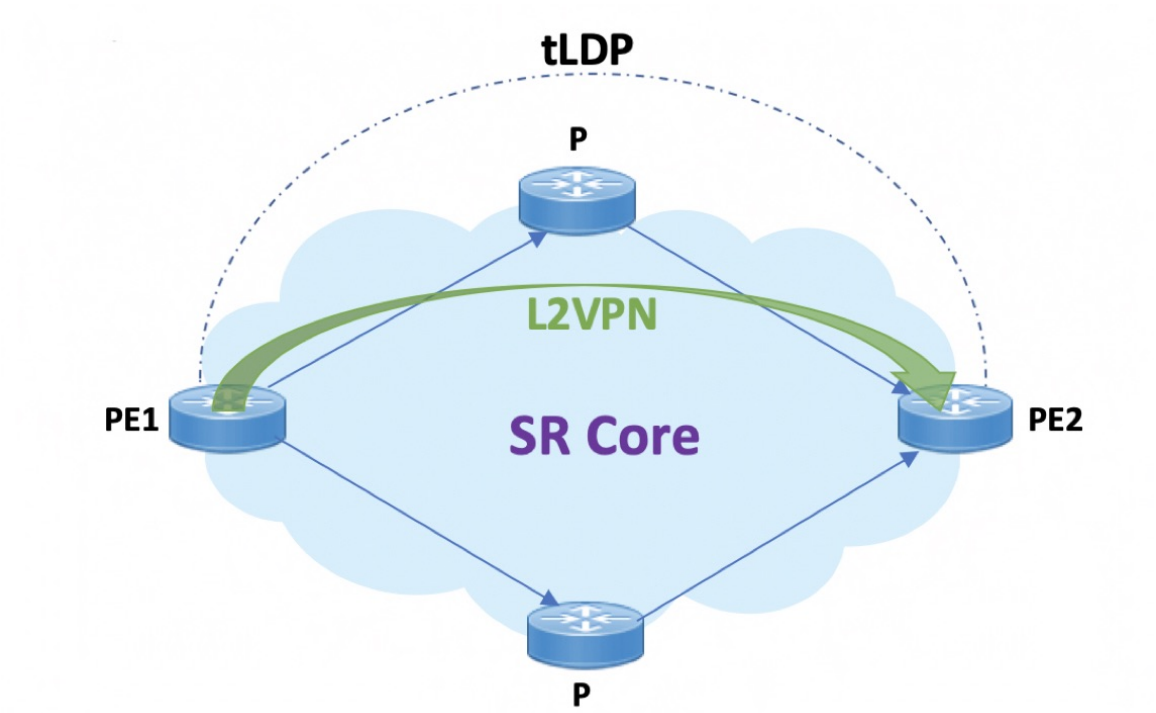
Capabilities Sent shows that **mldp-only** option disables all other advertisements.

Capabilities Received shows that **mldp-only** is enabled on peer PE2 too.

Disable Prefix-LSPs On An L2VPN/PW tLDP Session

A sample topology and relevant configurations are noted below.

Figure 9: L2VPN Xconnect Service Over Segment Routing



- The topology represents an L2VPN Xconnect service over a Segment Routing core setup.
- By default, Xconnect uses tLDP to signal service labels to remote PEs.
- By default, tLDP not only signals the service label, but also known (IPv4 and IPv6) label bindings to the tLDP peer, which is not required.
- The LDP SAC capabilities is an optional configuration enabled under LDP, and users can block IPv4 and IPv6 label bindings by applying configurations on PE1 and PE2.

Configuration

PE1 Configuration

Disable IPv4 prefix LSP binding advertisements on PE1:

```
PE1(config)# mpls ldp capabilities sac ipv4-disable
PE1(config)# commit
```

Disable IPv6-prefix LSP binding advertisements on PE1:

```
PE1(config)# mpls ldp capabilities sac ipv4-disable ipv6-disable
PE1(config)# commit
```

Note Whenever you disable a non-negotiated LDP application state on a router, you must include previously disabled non-negotiated LDP applications too, in the same command line. If not, the latest configuration overwrites the existing ones. You can see that ipv4-disable is added again, though it was already disabled.

PE2 Configuration

Enable SAC capability awareness on PE2, and make PE2 stop sending IPv4 prefix LSP binding advertisements to PE1:

```
PE2(config)#mpls ldp capabilities sac
PE2(config)#commit
```

Verification

On PE1, verify PE2's SAC capabilities:

```
PE1# show mpls ldp neighbor 198.51.100.1 detail

Peer LDP Identifier: 198.51.100.1:0
  TCP connection: 198.51.100.1:29132 - 192.0.2.1:646
  Graceful Restart: No
  Session Holdtime: 180 sec
  State: Oper; Msgs sent/rcvd: 14/14; Downstream-Unsolicited
  Up time: 00:03:30
  LDP Discovery Sources:
    IPv4: (1)
      Targeted Hello (192.0.2.1 -> 198.51.100.1, active)
    IPv6: (0)
  Addresses bound to this peer:
    IPv4: (3)
      203.0.113.1     209.165.201.1    10.0.0.1   198.51.100.1
      172.16.0.1
    IPv6: (0)
  Peer holdtime: 180 sec; KA interval: 60 sec; Peer state: Estab
  NSR: Disabled
  Clients: AToM
  Capabilities:
    Sent:
      0x508  (MP: Point-to-Multipoint (P2MP))
      0x509  (MP: Multipoint-to-Multipoint (MP2MP))
      0x50b  (Typed Wildcard FEC)
      0x50d  (State Advertisement Control)
        [ {IPv4-disable}  ] (length 1)
    Received:
      0x508  (MP: Point-to-Multipoint (P2MP))
      0x509  (MP: Multipoint-to-Multipoint (MP2MP))
      0x50b  (Typed Wildcard FEC)
      0x50d  (State Advertisement Control)
```

Capabilities Sent SAC capability **ipv4-disable** is sent, and local IPv4 label bindings are not generated.

Capabilities Received The peer (PE2) understands SAC capability and won't send its local IPv4 label bindings to local PE.

On PE1, verify SAC capabilities:

```
PE1# show mpls ldp capabilities detail

  Type     Description                                Owner
  ------   ----------------------------------------   ------------
  0x50b    Typed Wildcard FEC                         LDP
     Capability data: None

  0x3eff   Cisco IOS-XR                               LDP
     Capability data:
       Length: 12
       Desc  : [ host=PE1; platform=NCS 5500; release=07.01.01 ]
```

```
0x508    MP: Point-to-Multipoint (P2MP)             mLDP
   Capability data: None

0x509    MP: Multipoint-to-Multipoint (MP2MP)       mLDP
   Capability data: None

0x50d    State Advertisement Control                LDP
   Capability data:
     Length: 1
     Desc  : [ {IPv4-disable}  ]

0x703    P2MP PW                                    L2VPN-AToM
   Capability data: None
```

On PE1, verify that local and remote FEC bindings are removed.

```
PE1# show mpls ldp neighbor 198.51.100.1
Wed March 3 13:42:13.359 EDTs
```

MPLS traffic flow control for TTL and QoS propagation

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
MPLS traffic flow control for TTL and QoS propagation on MPLS push, pop, and penultimate nodes	Release 24.4.1	Introduced in this release on: NCS 5500 fixed port routers; NCS 5700 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards; NCS 5700 line cards [Mode: Compatibility; Native]). With this feature, the extended granular control capability for the incoming and outgoing MPLS traffic changes the behavior of the IP TTL and IP QoS DSCP propagation on the MPLS push, pop, and penultimate nodes. This ensures a reduced network latency, enhanced QoS management, and simplified network operations. This feature introduces these changes: **CLI:** • **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-push ttl** • **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop ttl-and-cos** • **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl-and-cos** **YANG Data Model:** New XPaths for `Cisco-IOS-XR-um-hw-module-profile-cfg.yang` (see Github, YANG Data Models Navigator).

MPLS traffic flow control for TTL and QoS propagation on penultimate node on NC57 line cards	Release 24.4.1	Introduced in this release on: NCS 5700 fixed port routers;NCS 5700 line cards [Mode: Native]. You can now control the incoming and outgoing MPLS and IP traffic granularly to change the behavior of IP Time-to- live (TTL) and IP QoS Differentiated Services Code Point (DSCP) propagation separately on the MPLS penultimate hop popping (PHP) node using the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** and **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos** configurations within the MPLS core network. This feature ensures reduced network latency, enhanced QoS management, and simplified network operations on the MPLS penultimate nodes. This feature introduces these changes: **CLI:** • **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** • **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos** **YANG Data Model:** New XPaths for `Cisco-IOS-XR-um-hw-module-profile-cfg.yang` (see Github, YANG Data Models Navigator).

In the existing behavior, MPLS traffic flows in uniform mode by default for the IP TTL and IP QoS propagation. You can enable the MPLS traffic to flow in pipe mode using the **mpls ip-ttl-propagate disable** configuration.

Starting with Cisco IOS XR Release 24.4.1, you can control the incoming and outgoing MPLS and IP traffic for the TTL and QoS DSCP propagation so that the traffic flow in Uniform mode on the MPLS push, pop, and PHP nodes.

MPLS traffic flow control for TTL and QoS propagation on the PHP node

You can control the MPLS traffic for TTL propagation on the PHP node using the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** configuration where the TTL propagation changes to uniform mode and QoS propagation is preserved in the pipe mode.

Similarly, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos** command to control the MPLS traffic for the QoS DSCP propagation on the PHP node so that the QoS propagation changes to uniform mode whereas the TTL propagation remains in the pipe mode.

The **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** and **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos**configurations are supported only on the NC57 line cards in native mode.

To control the MPLS traffic for both the TTL and QoS DSCP propagation on the PHP node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl-and-cos** configuration so that both the IP TTL and QoS propagation changes to uniform mode.

This table provides the difference in the MPLS traffic flow behavior for the TTL propagation and QoS propagation on the PHP node between the existing **mpls ip-ttl-propagate disable** configuration and the new **hw-module fib mpls ip-ttl-propagate-disable exclude** configuration.

Table 6: Existing and new behavior for MPLS TTL and QoS propagation on the PHP node

Existing configuration	New hardware module configuration	TTL/QoS/TTL and QoS behavior modification	
mpls ip-ttl-propagate disable	**hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop {ttl | cos | ttl-and-cos}**	**Old behaviour**	**New behavior**
Not configured	Not configured	Uniform	Uniform
Not configured	Configured	Uniform	Uniform
Configured	Not configured	Pipe	Pipe
Configured	Configured	Pipe	Uniform

MPLS traffic flow control for TTL and QoS propagation on the Pop node

You can control the MPLS traffic for TTL and QoS propagation on the pop node using the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop ttl-and-cos** configuration where both the TTL propagation and QoS propagation change to uniform mode on the MPLS pop node.

This table provides the difference in the MPLS traffic flow behavior for the TTL and QoS propagation on the pop node between the existing **mpls ip-ttl-propagate disable** configuration and the new **hw-module fib mpls ip-ttl-propagate-disable exclude** configuration.

Table 7: Existing and new behavior for MPLS TTL and QoS propagation on the Pop node

Existing configuration	New hardware module cnfiguration	TTL and QoS behavior modification	
mpls ip-ttl-propagate disable	**hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop ttl-and-cos**	**Old behaviour**	**New behavior**
Not configured	Not configured	Uniform	Uniform
Not configured	Configured	Uniform	Uniform
Configured	Not configured	Pipe	Pipe
Configured	Configured	Pipe	Uniform

MPLS traffic flow control for TTL and QoS propagation on the Push node

To control the MPLS traffic for the TTL propagation on the MPLS push node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-push ttl**. This configuration changes the TTL propagation to uniform mode.

This table provides the difference in the MPLS traffic behavior for the TTL propagation on the push node between the existing **mpls ip-ttl-propagate disable** configuration and the new **hw-module fib mpls ip-ttl-propagate-disable exclude** configuration.

Table 8: Existing and new behavior for MPLS TTL propagation on the push node

Existing configuration	New hardware module configuration	TTL behavior modification	
mpls ip-ttl-propagate disable	**hw-module fib mpls ip-ttl-propagate-disable exclude mpls-push ttl**	**Old behaviour**	**New behavior**
Not configured	Not configured	Uniform	Uniform
Not configured	Configured	Uniform	Uniform
Configured	Not configured	Pipe	Pipe
Configured	Configured	Pipe	Uniform

Restrictions

- The **hw-module fib mpls ip-ttl-propagate-disable exclude** configuration works only if the **mpls ip-ttl-propagate disable** command is configured.
- The **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** and **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos** configurations are supported only on the NC57 line cards in native mode.
- The **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl-and-cos** command cannot be configured along with either the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** configuration or the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos** configuration.

Configure MPLS Push, Pop, and PHP exclude operations

You can configure the **hw-module fib mpls ip-ttl-propagate-disable exclude** command for the TTL propagation and QoS DSCP propagation on the MPLS push, pop, and penultimate hop pop (PHP) nodes.

Configure MPLS push node for TTL propagation

Configuration:

To modify the MPLS traffic flow for TTL propagation in uniform mode on the MPLS push node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-push ttl** configuration.

This example shows the configuration that changes the IP TTL propagation to uniform mode with QoS remaining in the pipe mode on the MPLS push node.

```
RP/0/RP0/CPU0:router# config
RP/0/RP0/CPU0:router(config)# hw-module fib mpls ip-ttl-propagate-disable exclude mpls-push
 ttl
```

Make sure that you reload the router or all line cards on the router for the configuration to take effect.

Verification:

This example verifies whether the MPLS push node configuration for the TTL propagation is successful or not.

```
RP/0/RP0/CPU0:router# show run
hw-module fib mpls ip-ttl-propagate-disable exclude mpls-push ttl
```

This example shows whether the MPLS push node configuration for the TTL propagation is applied or not.

```
RP/0/RP0/CPU0:ios# show ofa objects global location 0/0/CPU0  | in ipttl_propagate_disable
Tue Oct 29 10:16:33.131 UTC
  ofa_bool_t ipttl_propagate_disable_mpls_push_exception => TRUE
  ofa_bool_t ipttl_propagate_disable_mpls_php_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_pop_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_cos_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_ttl_exception => FALSE
RP/0/RP0/CPU0:ios#
```

The value **TRUE** indicates that the configuration has been applied after the router reload.

Configure MPLS PHP node for TTL and Qos propagation

- **MPLS PHP node configuration for TTL propagation:**

  To modify the MPLS traffic flow for the IP TTL propagation in Uniform mode on the MPLS PHP node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl** command.

  This example shows the configuration that changes the IP TTL propagation to uniform mode with QoS remaining in the pipe mode on the MPLS PHP node.

  ```
  RP/0/RP0/CPU0:router# config
  RP/0/RP0/CPU0:router(config)# hw-module fib mpls ip-ttl-propagate-disable exclude
  mpls-pop-penultimate-hop ttl
  ```

Important Make sure that you reload the router or all line cards on the router for the configuration to take effect.

Verification:

This example verifies whether the MPLS PHP node configuration for the TTL propagation is successful or not.

```
RP/0/RP0/CPU0:router# show run
hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl
```

This example shows whether the MPLS PHP node configuration for the TTL propagation is applied or not.

```
RP/0/RP0/CPU0:ios# show ofa objects global location 0/0/CPU0  | in ipttl_propagate_disable
Tue Oct 29 10:16:33.131 UTC
  ofa_bool_t ipttl_propagate_disable_mpls_push_exception => FALSE
```

```
  ofa_bool_t ipttl_propagate_disable_mpls_php_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_pop_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_cos_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_ttl_exception => TRUE
RP/0/RP0/CPU0:ios#
```

The value **TRUE** indicates that the configuration has been applied after the router reload.

- **MPLS PHP node configuration for QoS propagation:**

  To modify the MPLS traffic flow for the QoS DSCP propagation in Uniform mode on the MPLS PHP node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos** command.

  This example shows the configuration that changes the QoS DSCP propagation to uniform mode with TTL remaining in the pipe mode on the MPLS PHP node.

```
RP/0/RP0/CPU0:router# config
RP/0/RP0/CPU0:router(config)# hw-module fib mpls ip-ttl-propagate-disable exclude
mpls-pop-penultimate-hop cos
```

Important Make sure that you reload the router or all line cards on the router for the configuration to take effect.

Verification:

This example verifies whether the MPLS PHP node configuration for the QoS propagation is successful or not.

```
RP/0/RP0/CPU0:router# show run
hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop cos
```

This example shows whether the MPLS PHP node configuration for the QoS propagation is applied or not.

```
RP/0/RP0/CPU0:ios# show ofa objects global location 0/0/CPU0  | in ipttl_propagate_disable
Tue Oct 29 10:16:33.131 UTC
  ofa_bool_t ipttl_propagate_disable_mpls_push_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_pop_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_cos_exception => TRUE
  ofa_bool_t ipttl_propagate_disable_mpls_php_ttl_exception => FALSE
RP/0/RP0/CPU0:ios#
```

The value **TRUE** indicates that the configuration has been applied after the router reload.

- **MPLS PHP node configuration for TTL and QoS propagation:**

  To modify the MPLS traffic flow for both the IP TTL and QoS DSCP propagation in uniform mode on the MPLS PHP node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl-and-cos** command.

  This example shows the configuration that changes the IP TTL and QoS DSCP propagation to uniform mode on the MPLS PHP node.

```
RP/0/RP0/CPU0:router# config
RP/0/RP0/CPU0:router(config)# hw-module fib mpls ip-ttl-propagate-disable exclude
mpls-pop-penultimate-hop ttl-and-cos
```

Important Make sure that you reload the router or all line cards on the router for the configuration to take effect.

Verification:

This example verifies whether the MPLS PHP node configuration for the TTL and QoS propagation is successful or not.

```
RP/0/RP0/CPU0:router# show run
hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl-and-cos
```

This example shows whether the MPLS PHP node configuration for the TTL and QoS propagation is applied or not.

```
RP/0/RP0/CPU0:ios# show ofa objects global location 0/0/CPU0  | in ipttl_propagate_disable
Tue Oct 29 10:16:33.131 UTC
  ofa_bool_t ipttl_propagate_disable_mpls_push_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_exception => TRUE
  ofa_bool_t ipttl_propagate_disable_mpls_pop_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_cos_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_ttl_exception => FALSE
RP/0/RP0/CPU0:ios#
```

The value **TRUE** indicates that the configuration has been applied after the router reload.

Configure MPLS pop node for TTL and QoS propagation

Configuration:

To modify the MPLS traffic flow for IP TTL and QoS DSCP propagation in uniform mode on the MPLS pop node, use the **hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop ttl-and-cos** configuration.

This example shows the configuration that changes both the IP TTL and QoS DSCP propagation to uniform mode on the MPLS pop node.

```
RP/0/RP0/CPU0:router# config
RP/0/RP0/CPU0:router(config)# hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop
 ttl-and-cos
```

Important Make sure that you reload the router or all line cards on the router for the configuration to take effect.

Verification:

This example verifies whether the MPLS pop node configuration for the TTL and QoS propagation is successful or not.

```
RP/0/RP0/CPU0:router# show run
hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop ttl-and-cos
```

This example shows whether the MPLS Pop node configuration for the TTL and QoS propagation is applied or not.

```
RP/0/RP0/CPU0:ios# show ofa objects global location 0/0/CPU0  | in ipttl_propagate_disable
Tue Oct 29 10:16:33.131 UTC
  ofa_bool_t ipttl_propagate_disable_mpls_push_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_pop_exception => TRUE
```

```
  ofa_bool_t ipttl_propagate_disable_mpls_php_cos_exception => FALSE
  ofa_bool_t ipttl_propagate_disable_mpls_php_ttl_exception => FALSE
RP/0/RP0/CPU0:ios#
```

The value **TRUE** indicates that the configuration has been applied after the router reload.

Granular TTL propagation and DSCP preservation controls on MPLS PHP nodes

Granular Time-to- live (TTL) propagation and Differentiated Services Code Point (DSCP) preservation controls on MPLS penultimate hop popping (PHP) nodes is a traffic engineering capability that

- allows selective propagation of TTL and IP QoS DSCP fields at the MPLS penultimate hop node,
- preserves original IP QoS DSCP values using Pipe mode, and
- inherits the TTL from the MPLS header using Uniform mode.

Table 9: Feature History Table

Feature Name	Release Information	Feature Description
Granular TTL propagation and DSCP preservation controls on MPLS PHP nodes	Release 25.3.1	Introduced in this release on: NCS 5500 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards). You can now customize TTL and QoS propagation independently for enhanced QoS management and simplified network operations on the MPLS penultimate nodes. This is made possible by configuring the penultimate hop node to preserve original IP QoS DSCP values using Pipe mode, while simultaneously inheriting the TTL from the MPLS header using Uniform mode. Previously, both fields followed the same mode and could not be configured separately. This feature introduces these changes: **CLI:** • **hw-module fib mpls php dscp-preserve** **YANG Data Models:** • New XPaths for `Cisco-IOS-XR-fia-hw-profile-cfg.yang` • New XPaths for `Cisco-IOS-XR-um-hw-module-profile-cfg.yang` (see Github, YANG Data Models Navigator).

Usage guidelines for configuring TTL propagation and DSCP preservation on MPLS PHP nodes

You must manually reload the chassis to activate the feature.

Restrictions for configuring TTL propagation and DSCP preservation on MPLS PHP nodes

Feature compatibility limitations

This feature cannot be configured together with

- Segment Routing SRv6 mode (**hw-module profile segment-routing srv6 mode micro-segment format**)
- MPLS extended DSCP preserve (**hw-module profile mpls-ext-dscp-preserve v4uc-enable** or **hw-module profile mpls-ext-dscp-preserve v6uc-enable**)
- MPLS IP TTL propagate disable exclusion at PHP (**hw-module fib mpls ip-ttl-propagate-disable exclude mpls-pop-penultimate-hop ttl-and-cos**)

Hardware and Platform Support

This feature is not supported on

- NCS 5700 fixed port routers, and
- NCS 5700 line cards [Mode: Compatibility; Native].

Behavioral change for TTL and QoS DSCP propagation on MPLS PHP nodes

The table shows how TTL and QoS DSCP propagation modes on MPLS PHP nodes change depending on whether the granular control feature is enabled.

Table 10: Behavioral change for TTL and QoS DSCP propagation on MPLS PHP nodes

When the feature is...	Then TTL behaves in...	And QoS behaves in...
not configured	Uniform mode	Uniform mode.
configured	Uniform mode	Pipe mode.

Configure granular TTL propagation and DSCP preservation controls on MPLS PHP nodes

Use this procedure to configure selective propagation of TTL and IP QoS DSCP fields at the MPLS PHP node.

Procedure

Step 1 Configure IP QoS DSCP value perseverance and TTL propagation.

Example:

```
Router# config
```

```
Router(config)# hw-module fib mpls php dscp-preserve
Router(config)# commit
```

You must manually reload the chassis to activate the feature.

Step 2 Verify the configuration is applied to the router.