NOKIA

Nokia Service Router Linux
7250 Interconnect Router
7730 Service Interconnect Router
Release 24.10

MPLS Guide

Table of contents

1 About this guide

This guide describes the services and provides configuration examples for the Multiprotocol Label Switching (MPLS) protocol used with the Nokia Service Router Linux (SR Linux).

This document is intended for users who plan to implement MPLS for SR Linux.

> **Note:**
> This manual covers the current release and may also contain some content to be released in later maintenance loads. See the *SR Linux Release Notes* for information about features supported in each load.
>
> Configuration and command outputs shown in this guide are examples only; actual displays may differ depending on supported functionality and user configuration.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

> **DANGER:** Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

> **WARNING:** Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

> **Caution:** Caution indicates that the described activity or situation may reduce your component or system performance.

> **Note:** Note provides additional operational information.

> **Tip:** Tip provides suggestions for use or best practices.

1.2 Conventions

Nokia SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- An open right-angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start** > **connect to**.
- A vertical bar (|) indicates a mutually exclusive argument.

- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 What's new

Topic	Location
MPLS null labels for static tunnels	Label values Configuring null label for static tunnels
MPLS entropy labels	MPLS entropy label
MPLS ECMP	MPLS ECMP (7730 SXR platforms)
LDP ECMP	LDP ECMP
Targeted LDP	Targeted LDP
Manual LDP RLFA	Manual LDP RLFA
Automatic LDP RLFA	Automatic LDP RLFA

3 Overview

The following provides a brief description of MPLS and LDP and lists the functionality supported by SR Linux in this release. It contains the following topics:

- About MPLS
- About LDP
- Supported functionality

3.1 About MPLS

Multiprotocol Label Switching (MPLS) is a label switching technology that provides the ability to set up connection-oriented paths over a connectionless IP network. MPLS facilitates network traffic flow and provides a mechanism to engineer network traffic patterns independently from routing tables. MPLS sets up a specific path for a sequence of packets. The packets are identified by a label stack inserted into each packet.

MPLS requires a set of procedures to enhance network-layer packets with label stacks, which then turns them into labeled packets. Routers that support MPLS are known as Label Switching Routers (LSRs). To transmit a labeled packet on a particular data link, an LSR must support the encoding technique.

In MPLS, packets can carry not only one label but a set of labels in a stack. An LSR can swap the label at the top of the stack, pop the label, or swap the label and push one or more labels into the stack. The processing of a labeled packet is completely independent of the level of hierarchy. The processing is always based on the top label, without regard for the possibility that some number of other labels may have been above it in the past, or that some number of other labels may be below it at present.

Supported platforms

MPLS is supported on the following platforms:

- 7250 IXR
- 7730 SXR

3.1.1 LSRs

LSRs perform different label switching functions based on their position in a Label Switched Path (LSP). The LSRs in an LSP do one of the following:

- The LSR at the or head-end of an LSP is the ingress label edge router (LER). The ingress LER can encapsulate packets with an MPLS header and forward them to the next router along the path. A point-to-point LSP can only have one ingress LER.

  The ingress LER is programmed to perform a label push operation. More specifically, it is programmed with an IP route that encapsulates matching IP packets using MPLS and forwards them to one or more next-hop routers. Each outgoing MPLS packet has a label stack (in the MPLS header), and the top

entry in each stack contains a label value pushed by the route lookup process that indicates the path to be followed.

- A transit LSR is an intermediate router in the LSP between the ingress and egress routers. Each transit LSR along the path is programmed to perform a label swap operation: the transit LSR is programmed with an MPLS route that matches the label value at the top of the label stack, pops that label stack entry, pushes a new top label and forwards the resulting MPLS packet to the next set of routers along the path.

- The penultimate LSR (the one before last) in the LSP can be programmed to perform a pop-and-forward operation, known as penultimate hop popping (PHP). In this case, the LSR is programmed with an MPLS route that matches the label value at the top of the label stack, pops that label stack entry, and forwards the resulting packet to the tail-end router. The packet sent by the PHP LSR to the egress LER may be the original IP payload packet, but the forwarding decision made by the PHP LSR is based only on the incoming top label value, not IP route lookup.

- The router at the tail-end of an LSP is the egress LER. The egress LER strips the MPLS encapsulation, which changes it from an MPLS packet to a data packet, and then forwards the packet to its destination using information in the forwarding table. Each point-to-point LSP can have only one egress router. The ingress and egress routers in an LSP cannot be the same router.

  If the penultimate LSR is just a normal transit LSR performing a label swap (no PHP), the egress LER must be programmed to perform the pop operation. In this case, the programmed MPLS route matches the label value at the top of the label stack, the egress LER pops that label entry and then does another lookup on the next header; usually this is an IP header and the packet is forwarded based on IP route lookup.

3.2 About LDP

Label Distribution Protocol (LDP) is a protocol used to distribute MPLS labels in non-traffic-engineered applications. LDP allows routers to establish LSPs through a network by mapping network-layer routing information directly to data link layer-switched paths.

An LSP is defined by the set of labels from the ingress LER to the egress LER. LDP associates a Forwarding Equivalence Class (FEC) with each LSP it establishes. A FEC is a collection of common actions associated with a class of packets. When an LSR assigns a label to a FEC, it must allow other LSRs in the path to know about the label. LDP helps to establish the LSP by providing a set of procedures that LSRs can use to distribute labels.

The FEC associated with an LSP specifies which packets are mapped to that LSP. LSPs are extended through a network as each LSR splices incoming labels for a FEC to the outgoing label assigned to the next-hop for the specific FEC.

LDP performs the label distribution only in MPLS environments. The LDP operation begins with a hello discovery process to find LDP peers in the network. LDP peers are two LSRs that use LDP to exchange label/FEC mapping information. An LDP session is created between LDP peers. A single LDP session allows each peer to learn the other's label mappings (LDP is bidirectional) and to exchange label binding information.

LDP signaling works with the MPLS label manager to manage the relationships between labels and the corresponding FEC. An MPLS label identifies a set of actions that the forwarding plane performs on an incoming packet before discarding it. The FEC is identified through the signaling protocol (in this case, LDP) and allocated a label. The mapping between the label and the FEC is communicated to the forwarding plane.

When an unlabeled packet ingresses the router, classification policies associate it with a FEC. The appropriate label is imposed on the packet, and the packet is forwarded.

Supported platforms

Link LDP is supported on the following platforms:

- 7250 IXR series
- 7730 SXR series

For IGP support with link LDP, SR Linux supports IS-IS or OSPF/OSPFv3 with IPv4 and IPv6 FECs.

3.3 LDP IPv6

SR Linux extends the LDP control plane and data plane to support LDP IPv6 adjacencies and sessions using 128-bit LSR IDs.

LDP IPv6 is supported for both link and targeted LDP.

The implementation allows for concurrent support of independent LDP IPv4 (32-bit LSR ID) and IPv6 (128-bit LSR ID) adjacencies and sessions between peer LSRs.

3.3.1 LDP operation in an IPv6 network

LDP IPv6 can be enabled on an SR Linux subinterface. The following figure shows the LDP adjacency and session over an IPv6 subinterface.

Figure 1: LDP adjacency and session over an IPv6 subinterface
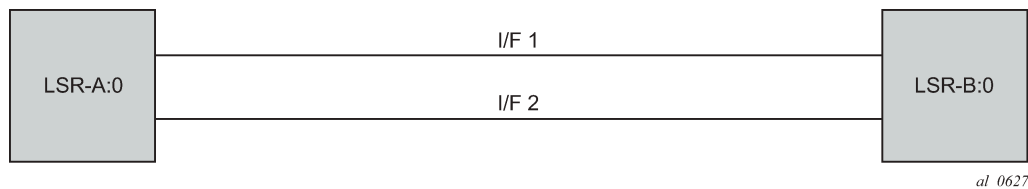
LSR-A and LSR-B have the following IPv6 LDP identifiers respectively:

- <LSR ID=A/128> : <label space id=0>
- <LSR ID=B/128> : <label space id=0>

By default, A/128 and B/128 use the system subinterface IPv6 address.

> **Note:** Although the LDP control plane can operate using only the IPv6 system address, you must configure the IPv4-formatted router ID for IS-IS to operate properly.

The following sections describe the behavior when LDP IPv6 is enabled on the subinterface.

3.3.2 Link LDP

The SR Linux LDP IPv6 implementation uses a 128-bit LSR ID as defined in RFC 5036.

The Hello adjacency is brought up using link Hello packets with a source IP address set to the subinterface link-local unicast address and a destination IP address set to the link-local multicast address FF02:0:0:0:0:0:0:2.

The transport address for the TCP connection, which is encoded in the Hello packet, is set to the LSR ID of the LSR by default.

3.3.3 FEC resolution

LDP advertises and withdraws all subinterface IPv6 addresses using the Address/Address-Withdraw message. Both the link-local unicast address and the configured global unicast addresses of an subinterface are advertised.

The LSR does not advertise a FEC for a link-local address and, if received, the LSR does not resolve it.

An IPv4 or IPv6 prefix FEC can be resolved to an LDP IPv6 subinterface in the same way it is resolved to an LDP IPv4 subinterface. The outgoing subinterface and next hop are looked up in the RTM cache. The next hop can be the link-local unicast address of the other side of the link or a global unicast address. The FEC is resolved to the LDP IPv6 subinterface of the downstream LDP IPv6 LSR that advertised the IPv4 or IPv6 address of the next hop.

3.4 Supported functionality

In the current release, SR Linux supports the following MPLS and LDP functionality:

MPLS

- Statically configured MPLS forwarding entries

- Configurable label range

- MPLS label manager that shares the MPLS label space among client applications that require MPLS labels

LDP

- LDPv4 implementation compliant with RFC 5036

- LDPv6 implementation *DM> compliant with RFC?*

- LDP support in the default network-instance only

- Label distribution using DU (downstream unsolicited), ordered control

- Platform label space only

- Configurable label range (dynamic, non-shared label-block)

- Support for overload TLV when label-block has no free entries

- Configurable timers (hello-interval, hello-holdtime, keepalive-interval)

- Ingress LER, transit LSR, and egress LER roles for /32 IPv4 or /128 IPv6 FECs

- Automatic FEC origination of the systemaddress

- /32 IPv4 FEC resolution using IGP routes, with longest prefix match option

- /128 IPv6 FEC resolution using IGP routes, with longest prefix match option

- ECMP support with configurable next-hop limit (up to 64)

- Automatic installation of all LDP /32 IPv4 and /128 IPv6 prefix FECs into TTM

- Per-peer configurable FEC limit

- Graceful restart helper capability

- BGP shortcuts over LDP tunnels

- Protocol debug/trace-options

- LDP-IGP synchronization

- Advertise address mapping message for primary IPv4 or IPv6 address of the adjacent subinterface only.

- Non-configurable capability advertisement in INITIALIZATION messages only claiming support for:

  – State advertisement control (SAC) with interest in IPv4 prefix FECs only

  – Fault tolerance (Graceful Restart)

  – Nokia overload TLV

  – Unrecognized notification

- Split-horizon support: A label-mapping message is not advertised to a peer if the FEC matches an address sent by that peer in an Address Mapping message.

4 Configuring MPLS

This chapter provides information about how MPLS functions on SR Linux and examples of common configuration tasks. It contains the following topics:

- MPLS label manager
- Static MPLS forwarding
- ACLs and MPLS traffic
- MPLS MTU
- TTL propagation and TTL expiry
- ICMP extensions for MPLS
- Label values
- MPLS ECMP
- Show reports for MPLS tunnel tables
- TTM preferences for supported tunnel types

4.1 MPLS label manager

SR Linux features an MPLS label manager process that shares the MPLS label space among client applications that require MPLS labels; these applications include static MPLS forwarding and LDP.

For a protocol such as LDP to become operational, it must be configured with a reference to a predefined range of labels, called a label block. A label block configuration includes a start-label value and an end-label value. When the label block is made available to an application, the application can use any label in the range between the start-label and end-label, inclusive of those values.

The MPLS label manager ensures there is no configuration overlap between the labels of two different label blocks.

4.1.1 Static and dynamic label blocks

A label block can be static or dynamic:

- A static label block is provided by the MPLS label manager to a client application when it is expected that the client application specifies the exact label value it wants to use with every label request.
- A dynamic label block is provided by the MPLS label manager to a client application when it is expected that the client application requests the next available label when it needs a new entry.

A label block can be configured as shared or dedicated. When a label block is configured to be shared, it allows the label block to be made available to multiple protocols. If a label block is not configured as shared, it is reserved for the exclusive use of one protocol.

The label block used by LDP must be configured as a dynamic, non-shared label block. For static MPLS routes, it is necessary to configure a static label block and reference the static label block in the static MPLS configuration.

4.1.2 Configuring label blocks

Procedure

To configure a static or dynamic label block, you specify the start and end label for the range.

Example

The following example configures a static and dynamic label block.

```
--{ * candidate shared default }--[  ]--
# info system mpls
    system {
        mpls {
            label-ranges {
                static s1 {
                    start-label 10001
                    end-label 11000
                }
                dynamic d1 {
                    start-label 11001
                    end-label 12000
                }
            }
        }
    }
```

4.1.3 Displaying label block information

Procedure

Use the **info from state** command to display information about the configured label blocks.

Example

```
--{ * candidate shared default }--[  ]--
# info from state system mpls label-ranges
    system {
        mpls {
            label-ranges {
                static {
                    name s1
                    start-label 10001
                    end-label 11000
                    allocated-labels 10
                    free-labels 990
                    status ready
                }
            }
        }
    }
```

4.2 Static MPLS forwarding

Statically configured MPLS forwarding entries allow MPLS-labeled packets to be forwarded along a specific path that may differ from the normal shortest path provided by the underlay routing protocol.

Static next-hop-groups used by IPv4 and IPv6 static routes of the default network-instance support MPLS next hops. An MPLS next-hop has a next-hop IP address and a list of MPLS labels (currently limited to 1). When an IP packet matches a static route pointing to a next-hop-group with MPLS next-hops, the packet is MPLS encapsulated according to the selected next-hop.

Static MPLS routes are supported in the default network-instance. Static MPLS routes control the way that transit LSRs, penultimate LSRs, and egress LERs handle received MPLS packets. Each static MPLS route matches a particular label value and causes that label value to be popped from the label stack when it appears as the top label in any received MPLS packet, on any interface. If the static MPLS route points to a next-hop-group with MPLS next-hops, the packet is forwarded to the selected next-hop with a swap operation; ECMP is supported if there are multiple MPLS next-hops. When the **pushed-mpls-label-stack** parameter for the MPLS next-hop specifies the IPv4 or IPv6 IMPLICIT_NULL label value, no new label is pushed and a PHP pop-and-forward operation is performed.

4.2.1 Configuring an ingress LER

Procedure

To configure an ingress LER, you specify the label to push to MPLS-encapsulated packets.

Example

The following example shows the default network-instance configured for an ingress LER, specifying the label to push to MPLS-encapsulated packets. In this example, multiple next-hops (ECMP) are specified in a next-hop group.

```
--{ * candidate shared default }--[  ]--
# info network-instance default next-hop-groups group srva_tora_ipv4
    network-instance default {
        next-hop-groups {
            group srva_tora_ipv4 {
                nexthop 0 {
                    ip-address 192.13.1.3
                    resolve false
                    pushed-mpls-label-stack [
                        544334
                    ]
                }
                nexthop 2 {
                    ip-address 192.13.1.3
                    resolve false
                    pushed-mpls-label-stack [
                        205679
                    ]
                }
            }
        }
    }
```

4.2.2 Configuring a transit LSR

Procedure

To configure a transit LSR, you specify a label block for MPLS and a next-hop to use for label-swap operations.

Example: Specify a static label block for MPLS

The following example shows the default network-instance configured as a transit LSR in the label switched path.

This example configures MPLS to use a static label block named s2. The static label block is configured with a start-label value and an end-label value. See Configuring label blocks for an example of a static label block configuration.

The example configures incoming MPLS transit traffic with label 1000 to use the next-hops in next-hop-group nhop_group_1 for label-swap operations.

```
--{ * candidate shared default }--[   ]--
# info network-instance default mpls
    network-instance default {
        mpls {
            static-label-block s2
            static-entry 1000 preference 100 {
                operation swap
                next-hop-group nhop_group_1
            }
        }
    }
```

Example: Specify next-hop for MPLS

The following example specifies the MPLS next-hop for nhop_group_1. Only one MPLS next-hop is supported per next-hop-group. In this example, the label for outgoing traffic to MPLS next-hop 192.35.1.5 is swapped to 1001.

```
--{ * candidate shared default }--[   ]--
# info network-instance default next-hop-groups group nhop_group_1
    network-instance default {
        next-hop-groups {
            group nhop_group_1 {
                nexthop 0 {
                    ip-address 192.35.1.5
                    resolve false
                    pushed-mpls-label-stack [
                        1001
                    ]
                }
            }
        }
    }
```

4.2.3 Configuring PHP

Procedure

To configure PHP, you configure MPLS to pop the label for outgoing traffic for a next-hop.

Example

The following example shows the default network-instance configured to perform PHP for the LSR. In this example, the setting for `pushed-mpls-label-stack` is `IMPLICIT_NULL`, which causes the label for outgoing traffic to MPLS next-hop 192.35.1.1 to be popped.

```
--{ * candidate shared default }--[  ]--
# info network-instance default next-hop-groups group nhop_group_2
    network-instance default {
        next-hop-groups {
            group nhop_group_2 {
                nexthop 0 {
                    ip-address 192.35.1.1
                    resolve false
                    pushed-mpls-label-stack [
                        IMPLICIT_NULL
                    ]
                }
            }
        }
    }
```

4.3 ACLs and MPLS traffic

Table 1: How MPLS traffic is handled by SR Linux ACLs lists how traffic along an MPLS datapath is evaluated by each type of ACL that can be configured on SR Linux. See the *SR Linux ACL and Policy-Based Routing Guide* for information about configuring ACLs on SR Linux.

Table 1: How MPLS traffic is handled by SR Linux ACLs

MPLS datapath	Evaluated by ingress ACL rules?	Evaluated by egress ACL rules?	Evaluated by CPU QoS entries?	Evaluated by IP CPM filter rules?	Evaluated by IP capture filter rules?
IP → MPLS (LER)	Yes	No	N/A	N/A	Yes
MPLS → MPLS (LSR)	Yes	No	N/A	N/A	No
MPLS → term (label TTL expiry)	Yes	N/A	No	No	No
MPLS → IP (PHP)	Yes	Yes	N/A	N/A	No
MPLS → IP (UHP)	Yes	Yes	N/A	N/A	Yes
MPLS → IP → term (IP address is local)	Yes	N/A	Yes	Yes	Yes

4.4 MPLS MTU

The MPLS MTU defines the maximum-sized MPLS packet that can be transmitted from a routed subinterface, including the size of the transmitted label stack (4 bytes * number of label stack entries). If an MPLS packet containing any payload exceeds this size, the packet is dropped.

SR Linux supports a system-wide default MPLS MTU value. If you configure a default MPLS MTU value, that value is programmed as the operational MPLS MTU on all IP interfaces of the system. The supported range for the default MPLS MTU is 1284 to 9496 bytes; default 1508 bytes.

In addition to the system-wide default MPLS MTU, you can configure a subinterface-level MPLS MTU, which applies to subinterfaces of type routed. The supported range for the subinterface-level MPLS MTU is 1284-9496 bytes. If no MPLS MTU is configured for a subinterface, the default MTU value is taken from the system-wide default MPLS MTU.

Each 7250 IXR IMM supports a maximum of four different MPLS MTU values. If a line card already has four different MPLS MTU values, and a fifth MPLS MTU value is configured for a subinterface on the same line card, the subinterface is brought down with the reason `mpls-mtu-resource-exceeded`.

4.4.1 Configuring the MPLS MTU

Procedure

You can configure a system-wide default MPLS MTU value, and you can configure a subinterface-level MPLS MTU that applies to subinterfaces of type routed. The supported range for the MPLS MTU is 1284 to 9496 bytes; default 1508 bytes. If no MPLS MTU is configured for a subinterface, the default MPLS MTU value is taken from the system-wide MPLS MTU.

Example: Configure a system-wide default MPLS MTU value

```
--{ * candidate shared default }--[  ]--
# info system mtu
    system {
        mtu {
            default-mpls-mtu 4096
        }
    }
```

Example: Configure an MPLS MTU for a routed subinterface

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1 subinterface 1
    interface ethernet-1/1 {
        subinterface 1 {
            type routed
            admin-state enable
            mpls-mtu 4096
            ipv4 {
                admin-state enable
                address 192.168.11.1/30 {
                }
            }
            ipv6 {
                admin-state enable
                address 2001:1::192:168:11:1/126 {
                }
```

```
            }
        }
    }
```

4.4.2 Displaying MPLS MTU information

Procedure

Use the **info from state** command to display MPLS MTU information.

Example: Display the system-wide default MPLS MTU value

```
--{ * candidate shared default }--[  ]--
# info from state system mtu default-mpls-mtu
    system {
        mtu {
            default-mpls-mtu 4096
        }
    }
```

Example: Display the MPLS MTU for a subinterface

```
--{ * candidate shared default }--[  ]--
# info from state interface ethernet-1/1 subinterface 1 mpls-mtu
    interface ethernet-1/1 {
        subinterface 1 {
            mpls-mtu 4096
        }
    }
```

4.5 TTL propagation and TTL expiry

SR Linux supports the Uniform model for TTL propagation as described RFC 3032 and RFC 3443. The Uniform model makes all the nodes that an LSP traverses visible to nodes outside the tunnel.

4.6 ICMP extensions for MPLS

SR Linux MPLS support includes ICMP extensions for transit LSRs and egress LERs.

- ICMP extensions for transit LSRs
- ICMP extensions for egress LERs

The ICMP extensions, defined in RFC 4950, are by default always enabled.

4.6.1 ICMP extensions for transit LSRs

When a transit LSR receives an MPLS packet that cannot be forwarded (for example, the label TTL has expired or the egress subinterface MPLS MTU was exceeded), the packet is extracted to the CPM, which

attempts to find an IP packet under the remaining label stack. If an IP packet is found, the CPM generates the appropriate error message, such as time-exceeded, destination-unreachable, or parameter-problem.

For time-exceeded and destination-unreachable messages only, the CPM generates a multipart ICMPv4/ICMPv6 time-exceeded message with the label stack object of RFC4950. This object encodes the entire MPLS label stack as it was received by the LSR that sends the ICMP message and at least 128 bytes of the original datagram (zero padded to this length if necessary).

4.6.1.1 Configuring ICMP tunneling

Procedure

ICMP tunneling is disabled by default. You can enable it for transit LSRs; on egress LERs, the setting for the **icmp-tunneling** option is not relevant.

If the **icmp-tunneling** option is disabled, all ICMP messages (including but not limited to multipath ICMP messages constructed according to RFC 4950) are injected into the default network instance for forwarding back to the source. The attempt to find a route to the source may be unsuccessful however.

If the **icmp-tunneling** option is enabled, all ICMP messages (including but not limited to multipath ICMP messages constructed according to RFC 4950) are injected in the forward direction of the LSP (that is, by re-adding the received label stack, resetting the MPLS TTL in all label stack entries to 255, and performing an ILM lookup on the top label). The source address of the ICMP message is an IP address of the LSR.

Example

The following example enables ICMP tunneling for a transit LSR.

```
--{ * candidate shared default }--[  ]--
# info network-instance default mpls icmp-tunneling
    network-instance default {
        mpls {
            icmp-tunneling true
        }
    }
```

4.6.2 ICMP extensions for egress LERs

If the egress LER receives an MPLS packet that cannot be forwarded (for example, the label TTL has expired, the IP TTL has expired, or the egress subinterface IP MTU was exceeded) the packet is extracted to the CPM with an indication of the network-instance associated with the last popped label. The CPM generates the appropriate ICMP error message, such as time-exceeded, destination-unreachable, and so on.

For time-exceeded and destination-unreachable messages only, the CPM generates a multipart ICMPv4/ICMPv6 time-exceeded message with the label stack object of RFC4950. This object encodes the entire MPLS label stack as it was received by the egress LER and at least 128 bytes of the original datagram (zero padded to this length if necessary).

All ICMP messages (including but not limited to multipath ICMP messages constructed according to RFC 4950) are injected into the network instance associated with the last popped label for forwarding back to the source.

If the egress LER receives an MPLS packet, pops the remaining label stack, and finds an IP packet that must be forwarded to a host address on a local subnet, the packet may be queued if there is no ARP entry

for the host address. If ARP cannot learn the MAC address of the host after about 3 seconds, the egress LER sends a destination-unreachable/host-unreachable message back to the source.

4.7 Label values

A packet traveling along an LSP is identified by its label, the 20-bit unsigned integer. The range is 0 to 1 048 575. Label values 0 to 31 are reserved and are defined as follows:

- A value of 0 represents the IPv4 explicit null label. It indicates that the label stack must be popped, and the packet forwarding must be based on the IPv4 header.

  The 7730 SXR implementation does not advertise an explicit null label, but can process an explicit null label if advertised. For LDP and SR-TE traffic, when the 7730 SXR receives a request from its peer for an explicit null push, the 7730 SXR honors the request and uses the explicit null label as the outermost transport label.

- A value of 1 represents the router alert label. The label value is legal anywhere in the label stack except at the bottom. When a received packet contains this label value at the top of the label stack, it is delivered to a local software module for processing. Packet forwarding is determined by the label beneath it in the stack. However, if the packet is further forwarded, the router alert label is pushed back onto the label stack before forwarding. The use of this label is analogous to the use of the router alert option in IP packets. Because this label cannot typically occur at the bottom of the stack, it is not associated with a specific network layer protocol.

  > **Note:** The router alert label can occur at the bottom of the stack when some operations, administration, and management (OAM) tools are used.

- A value of 3 represents the implicit null label. It is a label that a Label Switching Router (LSR) can assign and distribute, but which does not appear in the encapsulation. When an LSR would otherwise replace the label at the top of the stack with a new label, but the new label is implicit null, the LSR pops the stack instead of doing the replacement. Although this value may never appear in the encapsulation, it needs to be specified in the LDP protocol, so a value is reserved.

- A value of 7 represents the Entropy Label Indicator (ELI) which precedes the actual Entropy Label (EL) that carries the entropy value of the packet in the label stack.

- Values 4 to 6 and 8 to 31 are reserved for future use.

The router uses labels for MPLS, LDP, BGP Label Unicast, Segment Routing, as well as packet-based services such as VLL and VPLS.

To enable SR-MPLS features on SR Linux nodes, you must allocate a label range to the Segment Routing Global Block (SRGB). Other applications that require labels, such as LDP, can then be configured with a dynamic label block.

4.7.1 Configuring null label for static tunnels

About this task

For dynamic tunnels, SR Linux nodes honor reception of null label advertisement, meaning that the nodes push a null label when transmitting (or with implicit null, skip the push of the outermost tunnel label). However, SR Linux nodes do not support advertisement of null labels with dynamic tunnels.

For static tunnels, SR Linux supports the push of null labels using the **nexthop pushed-mpls-label-stack** command.

> ✏️ **Note:** Null labels are supported on 7250 IXR and 7730 SXR platforms.

Procedure

To enable **IPV4_EXPLICIT_NULL**, **IPV6_EXPLICIT_NULL**, or **IMPLICIT_NULL** for a next-hop group, use the **pushed-mpls-label-stack** command.

The explicit null label option indicates that the label stack must be popped, and the packet forwarding must be based on the IP header.

The implicit null label option allows an egress LER to receive MPLS packets from the previous hop without the outer LSP label. The operation of the previous hop is referred to as penultimate hop popping (PHP). This option is signaled by the egress LER to the previous hop during the FEC signaling by the LDP control protocol.

In addition, the egress LER can be configured to receive MPLS packet with the implicit null label on a static LSP. LDP must be shut down before changing the implicit null configuration option.

Example: Enable IPv4 explicit null label for next-hop group

The following example enables IPv4 explicit null labels for next-hop group `null-group`.

```
--{ candidate shared default }--[  ]--
# info network-instance default next-hop-groups group null-group
    network-instance default {
        next-hop-groups {
            group null-group {
                nexthop 1 {
                    resolve false
                    pushed-mpls-label-stack [
                        IPV4_EXPLICIT_NULL
                    ]
                }
            }
        }
    }
```

4.8 MPLS ECMP

The following sections describe MPLS ECMP options for SR Linux.

4.8.1 MPLS entropy label

> ✏️ **Note:** MPLS entropy label is supported on 7730 SXR platforms only.

The 7730 SXR supports MPLS entropy label, as described in RFC 6790. The entropy label provides greater granularity for load balancing on an LSR based on a hash label pushed by the iLER after deep inspection of various fields such as: source and destination IP address, source and destination MAC address, and so on, as applicable. In the absence of an entropy label, a typical LSR uses the MPLS label

stack for load balancing, which commonly yields to per-service load balancing. The entropy label also removes the need for the LSR to inspect the payload below the label stack and check for an IPv4 or IPv6 header to achieve better (per-flow) load-balancing.

The 7730 SXR supports entropy label on LDP, BGP-LU, SR-ISIS, and TE Policy SR-MPLS tunnels.

Note: The 7730 SXR does not support per-packet load balancing, but rather per-flow load balancing (essentially, per-flow entropy label).

The ability of a node to receive and process the entropy label for an LSP is signaled using capability signaling (referred to as entropy label capability).

Inserting an entropy label adds two labels in the MPLS label stack: the entropy label itself and the entropy label indicator (ELI). The entropy label is inserted directly below the tunnel label and closest to the service payload that has advertised entropy label capability (which may be above the bottom of the stack). The value of the entropy label is calculated based on a hash of the packet payload header. The ELI is a special-purpose MPLS label (value = 7) that indicates that the entropy label follows in the stack. The entropy label is always placed immediately below the tunnel label to which hashing applies.

The router inserts an entropy label on a tunnel that is entropy-label-capable when the service has entropy label enabled, even if an implicit or explicit null label has been signaled by the downstream LSR or LER. This ensures consistent behavior and that the entropy label value, as determined by the iLER, is maintained where a tunnel with an implicit null label is stitched at a downstream LSR.

If multiple transport tunnels have the **entropy-label** command enabled, the entropy label and ELI are inserted below the lowest transport label in the stack. The LSR parses the label stack to detect the ELI. The LSR hashes over the label stack and obtains values for different packets on the service, which spreads the service traffic across the different links, and multiple large services are not brought across the same LAG link member.

Entropy label capability is advertised at the tunnel level by the downstream LSR, which indicates the capability of the node to receive and process the entropy label.

Note: The 7730 SXR also supports control word with entropy label.

An LSR for LDP tunnels passes the entropy label capability from the downstream LSP segment to upstream peers. Earlier releases that do not support the entropy label functionality do not pass the entropy label capability to their peers.

The entropy label is typically inserted only if the downstream peer signals entropy label support. The router inserts only a single entropy label, even if multiple LSP labels exist in a label stack. The entropy label and ELI are inserted immediately below the innermost tunnel label for which the downstream peer LER is known to be entropy-label-capable. This ensures that the entropy label is preserved as far as possible along a path through the network.

The 7730 SXR supports inserting EL/ELI labels before the service label, but after the following transport labels:

- LDP
- SR-ISIS
- TE Policy SR-MPLS
- BGP-LU
    - LDP
    - SR-ISIS

- TE Policy SR-MPLS

- Implicit or explicit null

The 7730 SXR is also supported in a Label Switching Router role for the preceding cases based on hashing rules.

Entropy label insertion is configured using the **entropy-label** command. If entropy label insertion is configured, the MTU is automatically reduced to account for the overhead of the entropy label and ELI. This reduction occurs whether or not the LSP tunnel used by the service is entropy-label-capable.

In some cases, such as when entropy label is enabled for transport and control word is enabled, the entropy label requires the insertion of two additional labels in the label stack. This insertion can result in an unsupported label stack depth or large changes in the label stack depth during the lifetime of an LSP; for example, because a primary path with entropy label capability enabled is switched to a secondary path for which the far end has not signaled entropy label capability.

> **Note:** On the 7730 SXR, entropy labels do not count toward the label limit.

7730 SXR nodes operating in LSR and eLER roles support the processing of up to two entropy labels.

If an LSR is operating as an ASBR with NHS enabled, up to two EL/ELI label pairs are supported prior to the swapped label. If more than two EL/ELI pairs are present prior to the swapped label, the packet is dropped. Additional EL/ELI pairs are supported after the swap label.

eLER nodes support the termination of a maximum of two EL/ELIs. If more than two EL/ELI pairs are present in the label stack, excluding the payload, the packet is dropped.

On the LSR router, EL push at egress is dictated by the egress tunnel configuration. If the egress tunnel has entropy label enabled and the router popped an EL label, the router pushes an entropy label at egress. Other entropy labels in the label stack that are beneath the operated label do not influence this decision.

If the 7730 SXR does not pop an EL label prior to the operated label, and an EL is present deeper in the label stack, the router does not push a new entropy label, to optimize the label stack.

If more than one EL/ELI pair is in the incoming packet and one of the EL/ELI pairs is popped, the 7730 SXR entropy push operation is determined by the egress entropy configuration.

4.8.1.1 Ingress LER entropy labels

The 7730 SXR supports entropy labels per LSP, not per service.

The procedures at the ingress LER (iLER) are as specified in section 4.2 of RFC 6790. In general, the router inserts an entropy label in a packet if the downstream node for the LSP tunnel has signaled support for entropy labels and the entropy label is configured for the particular service.

RFC 6790 specifies that the iLER can insert several entropy labels in the label stack where the LSP hierarchy exists, one for each LSP in the hierarchy. However, this can result in unreasonably large label stacks. Therefore, when there are multiple LSPs in a hierarchy, the router only inserts a single entropy label and ELI pair within the innermost LSP label closest to the service payload that has advertised entropy label capability.

The following tables describe entropy label handling on iLER for tunnel over tunnel (ToT) use cases.

Table 2: Entropy label handling on iLER for ToT use cases (SR Policy)

Transport	Entropy Enabled?		Outcome
	TE Policy (LSP)	**SR-ISIS**	
TE Policy SR-MPLS Uncolored	Yes	Yes	EL/ELI pushed after transport label stack
	Yes	No	EL/ELI pushed after transport label stack
	No	Yes	EL/ELI pushed after SR-ISIS label ahead of TE Policy stack

Table 3: Entropy label handling on iLER for ToT use cases (BGP-LU over MPLS/SR-ISIS)

Transport	Entropy Enabled?		Outcome
	BGP	**MPLS/SR**	
BGP-LU over LDP/SR-ISIS	Yes	Yes	EL/ELI pushed after transport label stack
	Yes	No	EL/ELI pushed after transport label stack
	No	Yes	EL/ELI pushed after MPLS/SR-ISIS label ahead of BGP-LU label

Table 4: Entropy label handling on iLER for ToT use cases (BGP-LU over SR Policy)

Transport	Entropy Enabled?			Outcome
	BGP	**TE Policy (LSP)**	**SR-ISIS**	
BGP-LU over TE Policy (SR-MPLS)	Yes	Yes	Yes	EL/ELI pushed after transport label stack
	Yes	No	Yes	EL/ELI pushed after transport label stack
	Yes	No	No	EL/ELI pushed after transport label stack
	Yes	Yes	No	EL/ELI pushed after transport label stack
	No	Yes	Yes	EL/ELI pushed after **SR Label** stack
	No	Yes	No	EL/ELI pushed after **TE Policy** stack

Transport	Entropy Enabled?			Outcome
	BGP	**TE Policy (LSP)**	**SR-ISIS**	
	No	No	Yes	EL/ELI pushed after SR-ISIS

4.8.1.2 LSR entropy labels

Regardless of the load-balancing option configured on an LSR, if an entropy label is found in the label stack, the LSR exclusively uses the entropy label for load balancing without taking into account the MPLS label stack or any other field after BoS=1.

If PHP is requested by a next-hop LER, the LSR retains the entropy labels that are found immediately below the tunnel label that is to be popped. The system retains and uses the entropy label information as input to the local hash routine if an applicable LSR load-balancing mode is configured.

If the received label stack contains an entropy label prior to the swapped label, and the entropy label capability is enabled for the egress tunnel, the received and popped entropy label is pushed back onto the stack. This logic allows the hash generated by the iLER to be preserved end-to-end.

For example, when a packet is received with EL hash label and is operating as NHS for BGP-LU and the egress has multiple BGP next hops (ECMP) with multiple LSPs or NHLFEs to every BGP-LU next hop, the incoming ELI is used to hash traffic across multiple BGP-LU peers, followed by ECMP across parallel LSPs/NHLFEs to the identified peer.

ABR/ASBR (BGP-LU/BGP Prefix SID swap)

As an example, consider an ASBR with the following received label stack:

```
SVC + BGP-LU + EL/ELI + tunnel (LDP, SR, TE Policy, null)
```

In this case, the BGP-LU-enabled ASBR pops the tunnel labels and swaps the BGP-LU label. And if it is also configured for entropy label push against the egress tunnel, the ASBR transmits the following label stack:

```
SVC + BGP-LU (new) + EL/ELI + tunnel (new)
```

Figure 2: Inline BGP-LU RR
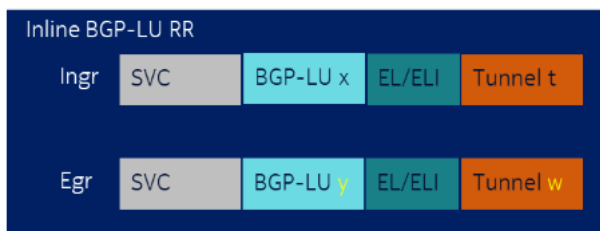
Note:

- If the 7730 SXR pops the entropy label and needs to push a new entropy label according to the egress tunnel configuration, it reuses the same EL label that it popped. Typically, the iLER has the most comprehensive access to the payload, which results in more detailed

identification of flows represented in the entropy label. The 7730 SXR reuses the incoming label to preserve the most detailed view of traffic flow.

- To avoid expanding the label stack depth, the 7730 SXR does not push more than one EL/ELI at egress even if multiple tunnels are configured for entropy label.

ABR/ASBR (full transport label termination, such as IP-VPN option-B)

As an additional example, consider an ABR with the following received label stack:

```
SVC + EL/ELI + BGP-LU + tunnel (LDP, SR, TE Policy, null)
```

In this case, the BGP-LU-enabled ABR pops the tunnel labels and swaps the SVC label, as follows:

- If the ABR is configured for EL/ELI push at BGP-LU level, the ABR transmits the following label stack (regardless of the tunnel-level entropy label configuration):

```
SVC (new) + EL/ELI + BGP-LU (new) + tunnel (new)
```

- If the ABR is not configured for EL/ELI push at BGP-LU level but is configured with entropy label for the tunnel, the ABR transmits the following label stack:

```
SVC (new) + BGP-LU (new) + EL/ELI + tunnel (new)
```

In either case, because the iLER computed EL is typically best to identify flows, and the iLER also has access to the payload, the same EL label is reused at egress.

Figure 3: IP-VPN Option B
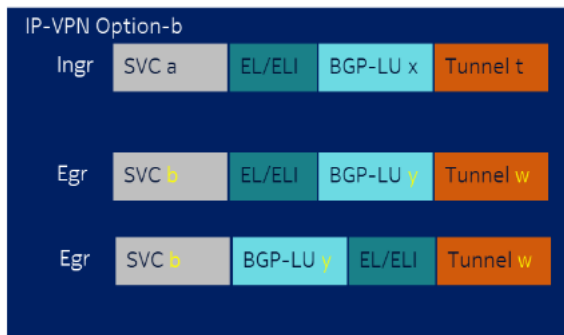
4.8.1.3 Egress LER entropy labels

At an egress LER (eLER), if an entropy label and ELI are detected in the label stack, both are popped and the packet is processed as normal. This occurs whether or not the system has signaled entropy label capability.

If an ELI is popped that has the bottom of stack (BoS) bit set, the system discards the packet and raises a trap, as described in section 4.1 of RFC 6790.

4.8.1.4 Entropy labels on self-generated traffic

Entropy label push for self-generated traffic such as **lsp-trace** packets is not supported. If an OAM packet is received with an entropy label, the entropy label is popped and the packet is handed off to the appropriate subsystem for handling.

4.8.1.5 Configuring MPLS entropy labels

About this task

The **entropy-label** command provides local control at the head-end of an LSP over whether the entropy label is inserted on an LSP by overriding the entropy label capability signaled from the far-end LER.

> **Note:**
> When the 7730 SXR is configured as the inline route reflector (RR) for an IP-VPN service, the entropy label is terminated at ingress. Regardless of the egress configuration, a new entropy label is not pushed.
>
> When the 7730 SXR is configured as the inline RR for a BGP-LU label termination, including swap with a new label, the entropy label is terminated. Regardless of the egress configuration, a new entropy label is not pushed.

Procedure

To enable MPLS entropy labels, use the **entropy-label** command for the following protocols using the specified contexts:

- BGP-LU: **network-instance protocols bgp bgp-label labeled-unicast entropy-label**
- LDP: **network-instance protocols ldp entropy-label**
- SR-ISIS: **network-instance protocols isis instance segment-routing mpls entropy-label**
- TE policy: **network-instance traffic-engineering-policies policy entropy-label**

Depending on the context, the following entropy label parameters are available:

- **advertise-capability**: when set to **true**, the router advertises the entropy label capability and includes the ELI/EL in the label stack only if the remote endpoint also signals support for the entropy label capability; when set to **false** (the default), no ELI/EL is included even if the endpoint signals support for the entropy label.
- **transmit**: when set to **enable**, the router includes the ELI/EL in the label stack whether or not the remote endpoint signals support for the entropy label capability (default: **disable**)

Example: Configure MPLS entropy labels for segment routing

The following shows a basic example of enabling entropy labels on SR-ISIS.

```
# info network-instance default protocols isis instance 1 segment-routing mpls
    network-instance default {
        protocols {
            isis {
                instance 1 {
                    segment-routing {
                        mpls {
                            entropy-label {
                                advertise-capability false
                                transmit enable
                            }
                        }
                    }
                }
            }
        }
    }
```

4.8.1.6 Displaying MPLS entropy label information from state

Procedure

To display state information for MPLS entropy labels, you can use the following **info from state** commands:

- **info from state network-instance protocols ldp [ipv4 | ipv6] bindings received-prefix-fec prefix-fec entropy-label-transmit**
- **info from state network-instance protocols ldp peers peer received-capabilities entropy-label-capability**
- **info from state network-instance route-table next-hop mpls entropy-label-transmit**
- **info from state network-instance traffic-engineering-policies policy segment-list entropy-label-transmit**
- **info from state network-instance traffic-engineering-policies policy-database sr-uncolored policy segment-list entropy-label-transmit**

4.8.2 MPLS ECMP (7730 SXR platforms)

For hash key generation on 7730 SXR platforms, SR Linux can parse the following seven unique header types:

- Ethernet headers:
  - The outermost Ethernet header on the wire counts as one unique header type.
  - The inner payload Ethernet header (EVPN IFF (interface-ful) case) counts as one unique header type.
- Up to two VLANs:
  - The inner VLAN counts as one unique header type.
  - The outer VLAN counts as one unique header type.
- Up to 14 labels
- Source/destination IP address
- Source/destination Layer 4 port
- Tunnel endpoint identifier (TEID) (TCP/UDP destination port 2123, 2152, or 3386)
- Entropy label

The hash key is generated based on fields within the first 128 bytes of the frame. Special labels, such as pseudowire control word (CW), are not included in the hash calculation.

The following table lists the fields used for hash key generation for each LSR role on 7730 SXR platforms. Note that the number of ECMP links in the group is used only for IP and MPLS ECMP; LAG hash does not include ECMP link count in the hash calculation.

Table 5: Fields used for hash key generation

Role	Traffic type	Fields used for hash key generation
iLER	Layer 3 subinterface traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • Source IP • Destination IP • Layer 4 source port • Layer 4 destination port • TEID
	Layer 2 subinterface traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • Source MAC • Destination MAC • Outermost VLAN
	IRB traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • Source MAC • Destination MAC • Outermost VLAN • Source port • Destination port • Source IP • Destination IP • TEID
LSR	Layer 2 payload (for example, MAC-VRF) traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • MPLS label stack • Source MAC • Destination MAC • Outermost VLAN

Role	Traffic type	Fields used for hash key generation
	Layer 2 and Layer 3 payload (for example, EVPN IFF) traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • MPLS label stack • Source MAC • Destination MAC • Outermost VLAN • Source IP • Destination IP • Source port • Destination port • TEID
	Layer 3 payload (for example, IP-VPN) traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • MPLS label stack • Source IP • Destination IP • Source port • Destination port • TEID
	ELI in stack	• System IP • Number of ECMP links in the group • Ingress subinterface index • Entropy label
eLER	Layer 3 subinterface traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • MPLS label stack • Source IP • Destination IP • Source port • Destination port

Role	Traffic type	Fields used for hash key generation
		• TEID
	Layer 2 subinterface traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • Source MAC • Destination MAC • Outermost VLAN
	IRB symmetric traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • Source IP • Destination IP • Source port • Destination port • TEID
	IRB asymmetric traffic	• System IP • Number of ECMP links in the group • Ingress subinterface index • Source MAC • Destination MAC • Outermost VLAN

TEID as hash key

GTP TEID plays a crucial role in mobile backhaul networks. User equipment (UE) such as mobile phones transmit a TEID value to identify data flows. In many cases, all flows from UEs between a radio and its controller can have the same source IP (for the radio), destination IP (for the packet gateway), source port, and destination port. To perform load balancing, the system must inspect deeper into the packet to use the TEID value as an input to the hash key.

When the IPv4/IPv6 UDP destination port is 2123 or 2152, the system assumes that the next header is a GTP header. To perform hashing with GTP TEID in the label stack, the system uses the following logic:

- If the next nibble following the MPLS BoS label is 4, the system assumes the payload is IPv4 and hashing is based on the MPLS label stack, IPv4, Layer 4, and TEID.

- If the next nibble following the MPLS BoS label is 6, the system assumes the payload is IPv6 and hashing is based on the MPLS label stack, IPv6, Layer 4, and TEID.

- If the next nibble following the MPLS BoS label is any value other than IPv4 or IPv6, the system assumes the payload is Ethernet and hashing is based on the MPLS label stack, Ethernet Layer 2, IP, and Layer 4.

Note: In the presence of an entropy label, the system uses only the entropy label for the hash calculation. The preceding TEID logic applies only to traffic without the entropy label.

iLER

For iLER Layer 2 subinterfaces, up to two VLANs are used for hash key generation. Up to 14 VLANs can be skipped for IP speculation to take effect.

LSR

If ELI is present, ELI is used for hashing. If ELI is not present, the MPLS label stack and Layer 2/3 properties are considered for hashing purposes. Label-only hash or label hash with Ethernet or IP headers can be configured to dictate the LSR hashing scope. In either case, if an entropy label is detected in the incoming label stack, it is used for hashing.

When no entropy label is present, up to 14 MPLS labels are used for hashing on the LSR:

*   If the first nibble is 4/6, hashing is based on source/destination IP and source/destination port, if destination TCP/UDP port = 2123 / 2152 / 3386, next-header is attempted to be hashed as described in TEID as hash key

*   If the first nibble is not 4/6, assume Ethernet, and hashing is based on source and destination MAC, Dot1q or QnQ tags based on ethertype, followed by hashing listed for the LSR role in Table 5: Fields used for hash key generation.

eLER

If ELI is present, ELI is used for hashing.

4.8.3 MPLS LSR ECMP (7250 IXR platforms)

SR Linux supports two options to perform MPLS ECMP packet hashing:

*   LSR label-only hash (default)
*   LSR label hash with Ethernet or IP headers (with L4 and TEID)

LSR label-only hash

By default, MPLS packet hashing at an LSR is based on the whole label stack, along with the incoming port and system IP address. With this label-only hash option, the system hashes the packet using up to 14 labels in the MPLS stack.

The system uses the net result to select which LSP next-hop to send the packet to using a modulo operation of the hash result with the number of next-hops.

This same result feeds to a second round of hashing if there is a LAG on the egress port where the selected LSP has its NHLFE programmed.

LSR label hash with Ethernet or IP headers (with L4 and TEID)

With the LSR label hash option with Ethernet or IP headers (with L4 and TEID), the datapath performs Ethernet speculation after the bottom of the label stack or when the maximum label stack depth is reached. Regardless of the outcome of the Ethernet speculation, the system also attempts IPv4/IPv6 speculation by checking the first nibble after the bottom of the label stack to include the following in the hash:

*   source and destination IP address

- UDP/TCP source and destination ports
- GPRS tunneling protocol (GTP) tunnel endpoint identifier (TEID) when the UDP or TCP port is 2123 or 2152

If Ethernet header speculation is successful, source and destination MAC are included in the hash along with up to two VLANs.

4.8.4 Configuring MPLS ECMP hashing options

Procedure

To configure MPLS ECMP hashing options on an LSR, use the **system load-balancing lsr-profile** command.

Example: Enable the label-only hash option

The following example enables the label-only hash option on an LSR.

```
--{ candidate shared default }--[  ]--
# system load-balancing lsr-profile label-stack
```

Example: Enable the LSR label hash option with Ethernet or IP headers (with L4 and TEID)

The following example enables the LSR label option with Ethernet or IP headers (with L4 and TEID) on an LSR.

```
--{ candidate shared default }--[  ]--
# system load-balancing lsr-profile label-eth-or-ip-l4-teid
```

4.9 Show reports for MPLS tunnel tables

You can issue a **show** command to display information about MPLS tunnel table entries. You can adjust the output of the report to filter by address type, encapsulation type, tunnel type, and destination prefix.

The following is an example of output from the **show** report.

Output example

```
--{ * candidate shared default }--[  ]--
# show network-instance default tunnel-table all
----------------------------------------------------------------------------------------------------
IPv4 tunnel table of network-instance "base"
----------------------------------------------------------------------------------------------------
--
+-------------+-------+------------+----------+-----+--------+------+----------------+----------+
| IPv4 Prefix | Encap | Tunnel Type| Tunnel ID| FIB | Metric | Pref | Next-hop (Type)| Next-hop |
|             | Type  |            |          |     |        |      |                |          |
+=============+=======+============+==========+=====+========+======+================+==========+
| 1.1.1.3/32  | mpls  | ldp        | 65629    | Y   | 2000   | 9    | 1.3.2.3 (mpls) | lag1.1   |
|             |       |            |          |     |        |      | 1.3.3.3 (mpls) | lag1.2   |
|             |       |            |          |     |        |      | 1.3.4.3 (mpls) | lag1.3   |
|             |       |            |          |     |        |      | 1.3.5.3 (mpls) | lag1.4   |
| 1.1.1.4/32  | mpls  | ldp        | 65631    | Y   | 2006   | 9    | 1.7.1.4 (mpls) | lag7.1   |
|             |       |            |          |     |        |      | 1.7.2.4 (mpls) | lag7.2   |
|             |       |            |          |     |        |      | 1.7.3.4 (mpls) | lag7.3   |
```

```
|             |       |         |         |     |      |      | 1.7.4.4 (mpls) | lag7.4   |
| 1.1.1.5/32  | mpls  | ldp     | 65630   | Y   | 2007 | 9    | 1.8.2.5 (mpls) | lag8.1   |
|             |       |         |         |     |      |      | 1.8.3.5 (mpls) | lag8.2   |
|             |       |         |         |     |      |      | 1.8.4.5 (mpls) | lag8.3   |
|             |       |         |         |     |      |      | 1.8.5.5 (mpls) | lag8.4   |
+-------------+-------+---------+---------+-----+------+------+----------------+----------+
```

4.10 TTM preferences for supported tunnel types

The following table summarizes the TTM preferences for all supported tunnel types on the 7250 IXR and 7730 SXR systems.

Table 6: TTM preferences for supported tunnel types

Tunnel type	TTM preference
Uncolored SR-MPLS TE-Policy	8
LDP	9
SR-ISIS	11
BGP (LU)	12

5 Configuring LDP

This chapter provides information about configuring Label Distribution Protocol (LDP) on SR Linux for both IPv4 and IPv6. It contains the following topics:

- Enabling LDP
- Configuring LDP neighbor discovery
- Configuring LDP peers
- Configuring a label block for LDP
- Configuring longest-prefix match for IPv4 and IPv6 FEC resolution
- Configuring load balancing over equal-cost paths
- Configuring graceful restart helper capability
- Configuring LDP-IGP synchronization
- Configuring static FEC (FEC originate)
- Overriding the LSR ID on an interface
- LDP FEC import and export policies
- LDP ECMP
- Targeted LDP
- Manual LDP RLFA
- Automatic LDP RLFA
- LSP ping and trace for LDP tunnels

5.1 Enabling LDP

Procedure

You must enable LDP for the protocol to be active. This procedure applies for both IPv4 and IPv6 LDP.

Example: Enable LDP

The following example administratively enables LDP for the default network instance.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp admin-state
    network-instance default {
        protocols {
            ldp {
                admin-state enable
            }
        }
    }
```

5.2 Configuring LDP neighbor discovery

Procedure

You can configure LDP neighbor discovery, which allows SR Linux to discover and connect to IPv4 and IPv6 LDP peers without manually specifying the peers. SR Linux supports basic LDP discovery for discovering LDP peers, using multicast UDP hello messages.

Example: Configure LDP neighbor discovery

The following example configures LDP neighbor discovery for a network instance and enables it on a subinterface for IPv4 and IPv6. The **hello-interval** parameter specifies the number of seconds between LDP link hello messages. The **hello-holdtime** parameter specifies how long the LDP link hello adjacency is maintained in the absence of link hello messages from the LDP neighbor.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp discovery
    network-instance default {
        protocols {
            ldp {
                discovery {
                    interfaces {
                        hello-holdtime 30
                        hello-interval 10
                        interface ethernet-1/1.1 {
                            ipv4 {
                                admin-state enable
                            }
                            ipv6 {
                                admin-state enable
                            }
                        }
                    }
                }
            }
        }
    }
```

5.3 Configuring LDP peers

Procedure

You can configure settings that apply to connections between SR Linux and IPv4 and IPv6 LDP peers, including session keepalive parameters. For individual LDP peers, you can configure the maximum number of FEC-label bindings that can be accepted by the peer.

If LDP receives a FEC-label binding from a peer that puts the number of received FECs from this peer at the configured FEC limit, the peer is put into overload. If the peer advertised the Nokia-overload capability (if it is another SR Linux router or an SR OS device) then the overload TLV is transmitted to the peer and the peer stops sending any further FEC-label bindings. If the peer did not advertise the Nokia-overload capability, then no overload TLV is sent to the peer. In either case, the received FEC-label binding is deleted.

Example: Configure LDP peers

The following example configures settings for IPv4 and IPv6 LDP peers:

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp peers
    network-instance default {
        protocols {
            ldp {
                peers {
                    session-keepalive-holdtime 240
                    session-keepalive-interval 90
                    peer 10.1.1.1 label-space-id 0 {
                        fec-limit 1024
                    }
                    peer 2001:db8::0a01:0101 label-space-id 0 {
                        fec-limit 1024
                    }
                }
            }
        }
    }
```

In this example, the **session-keepalive-holdtime** parameter specifies the number of seconds an LDP session can remain inactive (no LDP packets received from the peer) before the LDP session is terminated and the corresponding TCP session closed. The **session-keepalive-interval** parameter specifies the number of seconds between LDP keepalive messages. SR Linux sends LDP keepalive messages at this interval only when no other LDP packets are transmitted over the LDP session.

For an individual LDP peer, indicated by its LSR ID and label space ID, a FEC limit is specified. SR Linux deletes FEC-label bindings received from this peer beyond this limit.

5.4 Configuring a label block for LDP

Procedure

To configure LDP, you must specify a reference to a predefined range of labels, called a label block. A label block configuration includes a start-label value and an end-label value. LDP uses labels in the range between the start-label and end-label in the label block.

A label block can be static or dynamic. See Static and dynamic label blocks for information about each type of label block and how to configure them. LDP requires a dynamic, non-shared label block.

Example: Configure dynamic LDP label block

The following example configures LDP to use a dynamic label block named d1. The dynamic label block is configured with a start-label value and an end-label value. See Configuring label blocks for an example of a dynamic label block configuration.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp dynamic-label-block
    network-instance default {
        protocols {
            ldp {
                dynamic-label-block d1
            }
        }
```

```
}
```

5.5 Configuring longest-prefix match for IPv4 and IPv6 FEC resolution

Procedure

By default, SR Linux supports /32 IPv4 and /128 IPv6 FEC resolution using IGP routes. You can optionally enable longest-prefix match for IPv4 and IPv6 FEC resolution. When this is enabled, IPv4 and IPv6 prefix FECs can be resolved by less-specific routes in the route table, as long as the prefix bits of the route match the prefix bits of the FEC. The IP route with the longest prefix match is the route that is used to resolve the FEC.

Example: Configure longest-prefix match for IPv4 and IPv6 FEC resolution

The following example enables longest-prefix match for IPv4 and IPv6 FEC resolution.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp fec-resolution
    network-instance default {
        protocols {
            ldp {
                fec-resolution {
                    longest-prefix true
                }
            }
        }
    }
```

5.6 Configuring load balancing over equal-cost paths

Procedure

ECMP support for LDP on SR Linux performs load balancing for LDP-based tunnels by using multiple outgoing next-hops for an IP prefix on ingress and transit LSRs. You can specify the maximum number of next-hops (up to 64) to be used for load balancing toward a specific FEC.

Example: Configure load balancing for LDP

The following example configures the maximum number of next-hops that SR Linux can use for load balancing toward an IPv4 or IPv6 FEC.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp multipath
    network-instance default {
        protocols {
            ldp {
                multipath {
                    max-paths 64
                }
            }
        }
    }
```

5.7 Configuring graceful restart helper capability

Procedure

Graceful restart allows a router that has restarted its control plane but maintained its forwarding state to restore LDP with minimal disruption.

To do this, the router relies on LDP peers, which have also been configured for graceful restart, to maintain forwarding state while the router restarts. LDP peers configured in this way are known as graceful restart helpers.

You can configure SR Linux to operate as a graceful restart helper for LDP. When the graceful restart helper capability is enabled, SR Linux advertises to its LDP peers by carrying the fault tolerant (FT) session TLV in the LDP initialization message, which assists the LDP peer to preserve its LDP forwarding state across the restart.

Example: Configure graceful restart

The following example enables the graceful restart helper capability for LDP for both IPv4 and IPv6 peers.

```
--{ * candidate shared default }--[   ]--
# info network-instance default protocols ldp graceful--restart
    network-instance default {
        protocols {
            ldp {
                graceful-restart {
                    helper-enable true
                    max-reconnect-time 180
                    max-recovery-time 240
                }
            }
        }
    }
```

In this example, the **max-reconnect-time** parameter specifies the number of seconds the SR Linux waits for the remote LDP peer to reconnect after an LDP communication failure. The **max-recovery-time** parameter specifies the number of seconds the SR Linux router preserves its MPLS forwarding state after receiving the LDP initialization message from the restarted LDP peer.

5.8 Configuring LDP-IGP synchronization

Procedure

You can configure synchronization between LDP and IPv4 or IPv6 interior gateway protocols (IGPs). LDP-IGP synchronization is supported for IS-IS.

When LDP-IGP synchronization is configured, LDP notifies the IGP to advertise the maximum cost for a link in the following scenarios: when the LDP hello adjacency goes down, when the LDP session goes down, or when LDP is not configured on an interface.

The following apply when LDP-IGP synchronization is configured:

• If a session goes down, the IGP increases the metric of the corresponding interface to max-metric.

- When the LDP adjacency is reestablished, the IGP starts a hold-down timer (default 60 seconds). When this timer expires, the IGP restores the normal metric, if it has not been restored already.

- When LDP informs the IGP that all label-FEC mappings have been received from the peer, the IGP can be configured to immediately restore the normal metric, even if time remains on the hold-down timer.

LDP-IGP synchronization does not take place on LAN interfaces unless the IGP has a point-to-point connection over the LAN, and does not take place on IGP passive interfaces.

Example: Configure LDP-IGP synchronization

The following example configures LDP-IGP synchronization for an IS-IS instance.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols isis instance i1 ldp-synchronization
    network-instance default {
        protocols {
            isis {
                instance i1 {
                    ldp-synchronization {
                        hold-down-timer 120
                        end-of-lib true
                    }
                }
            }
        }
    }
```

In this example, if the LDP adjacency goes down, the IGP increases the metric of the corresponding interface to max-metric. If the adjacency is subsequently reestablished, the IGP waits for the amount of time configured with the **hold-down-timer** parameter before restoring the normal metric.

When the **end-of-lib** parameter is set to **true**, it causes the IGP to restore the normal metric when all label-FEC mappings have been received from the peer, even if time remains in the hold-down timer.

5.9 Configuring static FEC (FEC originate)

About this task

Static FEC (also known as FEC originate) triggers a label for prefix announcement without a requirement to enable LDP on a subinterface or to receive a label from a neighbor. Network security requirements may dictate that prefixes advertised using LDP must not be directly associated with a subinterface or system IP address. Static FEC allows the router to advertise these prefixes and their labels to peers as LDP FECs to provide reachability to the desired IP addresses or subnets without explicitly using a subinterface address. The router can advertise a FEC with a pop action.

A FEC can be added to the LDP IP prefix database with a specific label operation on the node. The only permitted operation is pop (the **swap** parameter must be set to **false**).

A route-table entry is required for a FEC to be advertised. Static FEC is supported for both IPv4 and IPv6 FECs.

Procedure

To configure static FEC, use the **static-fec** command under the **network-instance protocols ldp** context.

Example: Configure static FEC

```
--{ + candidate shared default }--[   ]--
# info network-instance default protocols ldp static-fec 10.10.10.2/32
    network-instance default {
        protocols {
            ldp {
                static-fec 10.10.10.2/32 {
                    swap false
                }
            }
        }
    }
```

5.10 Overriding the LSR ID on an interface

About this task

For security concerns, it may be beneficial to overwrite the default LSR ID with a different LSR ID for link or targeted LDP sessions. The following LSR IDs can be overwritten:

* IPv4 I-LDP: local subinterface IPv4 address
* IPv6 I-LDP: local subinterface IPv4 or IPv6 address, in one of the following combinations:
  – the subinterface IPv4 address as the 32-bit LSR-ID and the subinterface IPv6 address as the transport connection address
  – the subinterface IPv6 address as both a 128-bit LSR-ID and transport connection address
* IPv4 T-LDP: IPv4 loopback or any IPv4 LDP subinterface
* IPv6 T-LDP: IPv4 loopback, IPv6 loopack, IPv4 LDP subinterface, or IPv6 LDP subinterface

Note that a loopback interface cannot be used with the 32-bit format.

To change the value of the LSR ID on an interface to the local interface IP address, use the **override-lsr-id** option. When **override-lsr-id** is enabled, the transport address for the LDP session and the source IP address of the Hello messages is updated to the interface IP address.

In IPv6 networks, either the IPv4 or IPv6 interface address can override the IPv6 LSR ID.

Procedure

To override the value of the LSR ID on an interface, use the following commands under the **network-instance** context:

* **IPv4**:

  protocols ldp discovery interfaces interface *<name>* **ipv4 override-lsr-id local-subinterface ipv4**

* **IPv6**:

  protocols ldp discovery interfaces interface *<name>* **ipv6 override-lsr-id local-subinterface [ipv4 | ipv6]**

Example: Configure the LSR ID

```
--{ * candidate shared default }--[   ]--
# info network-instance default protocols ldp discovery interfaces interface ethernet-1/
1.1
```

```
        network-instance default {
            protocols {
                ldp {
                    discovery {
                        interfaces {
                            interface ethernet-1/1.1 {
                                ipv4 {
                                    override-lsr-id {
                                        local-subinterface ipv4
                                    }
                                }
                                ipv6 {
                                    override-lsr-id {
                                        local-subinterface ipv6
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
```

5.11 LDP FEC import and export policies

SR Linux supports FEC prefix import and export policies for LDP, which provide filtering of both inbound and outbound LDP label bindings.

FEC prefix export policy

A FEC prefix export policy controls the set of LDP prefixes and associated LDP label bindings that a router advertises to its LDP peers. By default, the router advertises local label bindings for only the system address, but propagates all FECs that are received from neighbors. The export policy can also be configured to advertise local interface FECs.

The export policy can accept or reject label bindings for advertisement to the LDP peers.

When applied globally, LDP export policies apply to FECs advertised to all neighbors. To control the propagation of FECs to a specific LDP neighbor, you can apply an LDP export prefix policy to the specified peer.

Note: The export policy does not support blocking of static FECs.

FEC prefix import policy

A FEC prefix import policy controls the set of LDP prefixes and associated LDP label bindings received from other LDP peers that a router accepts. The router redistributes all accepted LDP prefixes that it receives from its neighbors to other LDP peers (unless rejected by the FEC prefix export policy).

The import policy can accept or reject label bindings received from LDP peers. By default, the router imports all FEC prefixes from its LDP peers.

When applied globally, LDP import policies apply to FECs received from all neighbors. To control the import of FECs from a specific LDP neighbor, you can apply an LDP import prefix policy to the specified peer.

Routing policies

The filtering of label bindings in LDP FEC import and export policies is based on prefix lists that are defined using routing policies.

5.11.1 Configuring routing policies for LDP FEC import and export policies

Procedure

To configure routing policies for LDP FEC import and export policies, use the **routing-policy** command.

Example: Configure routing policy for global LDP FEC import and export

The following shows an example configuration of global LDP FEC import and export policies, defining match rules to accept an import prefix set, and reject an export prefix set.

```
--{ * candidate shared default }--[ ]--
# info routing-policy
    routing-policy {
        prefix-set export-prefix-set-test {
            prefix 10.1.1.2/32 mask-length-range exact {
            }
            prefix 10.1.1.3/32 mask-length-range exact {
            }
        }
        prefix-set import-prefix-set-test {
            prefix 10.1.1.1/32 mask-length-range exact {
            }
            prefix 10.1.1.2/32 mask-length-range exact {
            }
            prefix 10.1.1.3/32 mask-length-range exact {
            }
        }
        policy export-fec-test {
            default-action {
                policy-result accept
            }
            statement export-statement-test {
                match {
                    prefix-set export-prefix-set-test
                }
                action {
                    policy-result reject
                }
            }
        }
        policy import-fec-test {
            default-action {
                policy-result accept
            }
            statement import-statement-test {
                match {
                    prefix-set import-prefix-set-test
                }
                action {
                    policy-result accept
                }
            }
        }
    }
```

Example: Configure routing policy for peer LDP FEC import and export

The following shows an example configuration of peer LDP FEC import and export policies, defining match rules to reject an import prefix set and an export prefix set for the specified peer.

```
--{ * candidate shared default }--[ ]--
# info routing-policy
    routing-policy {
        prefix-set peer-export-prefix-test {
            prefix 10.1.1.4/32 mask-length-range exact {
            }
        }
        prefix-set peer-import-prefix-test {
            prefix 10.1.1.5/32 mask-length-range exact {
            }
        }
        policy peer-export-test {
            statement peer-export-statement-test {
                match {
                    prefix-set export-prefix-set-test
                }
                action {
                    policy-result reject
                }
            }
        }
        policy peer-import-test {
            statement peer-import-statement-test {
                match {
                    prefix-set import-prefix-set-test
                }
                action {
                    policy-result reject
                }
            }
        }
    }
```

5.11.2 Applying global LDP FEC import and export policies

Procedure

Use the **ldp import-prefix-policy** and **ldp export-prefix-policy** commands to apply global LDP FEC import and export policies.

The following example applies global export and import policies to LDP FECs.

Example: Apply global LDP import and export policies

```
--{ +* candidate shared default }--[  ]--
# info network-instance default protocols ldp
    network-instance default {
        protocols {
            ldp {
                export-prefix-policy export-fec-test
                import-prefix-policy import-fec-test
            }
        }
    }
```

5.11.3 Applying per-peer LDP FEC import and export policies

Procedure

Use the **import-prefix-policy** and **export-prefix-policy** commands under the **ldp peers peer** context to apply per-peer LDP FEC import and export policies.

The following example applies LDP FEC export and import policies to LDP peer 10.10.10.1.

Example: Apply global LDP import and export policies

```
--{ +* candidate shared default }--[  ]--
# info network-instance default protocols ldp peers peer 10.10.10.1 label-space-id 1
    network-instance default {
        protocols {
            ldp {
                peers {
                    peer 10.10.10.1 label-space-id 1 {
                        export-prefix-policy peer-export-test
                        import-prefix-policy peer-import-test
                    }
                }
            }
        }
    }
```

5.12 LDP ECMP

LDP ECMP performs load balancing for LDP-based LSPs by using multiple outgoing next hops for an IP prefix on ingress and transit LSRs.

An LSR that has multiple equal cost paths to a specific IP prefix can receive an LDP label mapping for this prefix from each of the downstream next-hop peers. Because the LDP implementation uses the liberal label retention mode, it retains all the labels for an IP prefix received from multiple next-hop peers.

Without LDP ECMP support, only one of these next-hop peers is selected and installed in the forwarding plane. The algorithm used to select the next-hop peer looks up the route information obtained from the RTM for this prefix and finds the first valid LDP next-hop peer (for example, the first neighbor in the RTM entry from which a label mapping was received). If the outgoing label to the installed next hop is no longer valid (for example, the session to the peer is lost or the peer withdraws the label), a new valid LDP next-hop peer is selected out of the existing next-hop peers, and LDP reprograms the forwarding plane to use the label sent by this peer.

With ECMP support, all the valid LDP next-hop peers (peers that sent a label mapping for an IP prefix) are installed in the forwarding plane. In both cases, an ingress LER and a transit LSR, an ingress label is mapped to the next hops that are in the RTM and from which a valid mapping label has been received. The forwarding plane then uses an internal hashing algorithm to determine how the traffic is distributed among these multiple next hops, assigning each flow to a specific next hop.

SR Linux supports two options to perform ECMP packet hashing:

- LSR label-only hash (default)
- LSR label hash with Ethernet or IP headers (with L4 and TEID)

See MPLS LSR ECMP (7250 IXR platforms) for information about these options.

Support platforms

LDP ECMP is supported on 7730 SXR platforms only.

5.13 Targeted LDP

Note: Targeted LDP is supported on 7730 SXR platforms only.

The 7730 SXR platforms support targeted LDP (T-LDP) for both IPv4 and IPv6, using IS-IS or OSPF/OSPFv3 as the IGP. While LDP sessions are typically established between two directly connected peers, T-LDP allows LDP sessions to be established between non-adjacent peers using LDP extended discovery. The source and destination addresses of targeted Hello packets are the LSR-IDs of the remote LDP peers.

In SR Linux, T-LDP sessions use the system IP address as the LSR-ID by default. You can optionally configure the **override-lsr-id** parameter on the targeted session to change the value of the LSR-ID to either an LDP-enabled local subinterface or to a loopback subinterface. The transport address for the LDP session and the source IP address of the targeted Hello message are then updated to the new LSR-ID value.

With IPv4 T-LDP, the **override-lsr-id** parameter can update the LSR-ID to an IPv4 subinterface or loopback subinterface. With IPv6 T-LDP, the **override-lsr-id** parameter can update the LSR-ID to an IPv4 or IPv6 subinterface or loopback subinterface.

The 7730 SXR supports multiple ways of establishing a targeted Hello adjacency to a peer LSR:

- Configuration of the peer with targeted session parameters either explicitly configured for the peer or inherited from the global LDP targeted session. The explicit peer settings override the top level parameters shared by all targeted peers. Some parameters only exist in the global context and their value is always inherited by all targeted peers regardless of which event triggered the targeted adjacency.

- Configuration of a pseudowire (FEC 128) using a **pw-tunnel** to the peer LSR in a virtual private wire service (VPWS) or MAC-VRF network instance. The targeted session parameter values are taken from the global context.

As the preceding triggering events can occur simultaneously or in any arbitrary order, the LDP code implements a priority handling mechanism to determine which event overrides the active targeted session parameters. The overriding trigger becomes the owner of the targeted adjacency to a specific peer and is displayed using the **show network-instance default protocols ldp neighbor** command.

The following table summarizes the triggering events and the associated priority.

Table 7: Triggering events and the associated priority

Triggering event	Automatic creation of targeted Hello adjacency	Active targeted adjacency parameter override priority
Manual configuration of peer parameters (creator= manual)	Yes	1
Pseudowire and pw-tunnel configuration in a network instance (creator=pw manager)	Yes	3

Any parameter value change to an active targeted Hello adjacency caused by any of the triggering events in the preceding table is performed immediately by having LDP send a Hello message with the new parameters to the peer without waiting for the next scheduled time for the Hello message. This functionality allows the peer to adjust its local state machine immediately and maintains both the Hello adjacency and the LDP session in the up state. The only exceptions are the following:

- The triggering event caused a change to the **override-lsr-id** parameter value. The Hello adjacency is brought down and causes the LDP session to be brought down if this is the last Hello adjacency associated with the session. A new Hello adjacency and LDP session is established to the peer using the new value of the local LSR ID.

- The triggering event caused the targeted peer **admin-state** option to be enabled. The Hello adjacency is brought down and causes the LDP session to be brought down if this is the last Hello adjacency associated with the session.

The value of any LDP parameter that is specific to the peer LDP/TCP session is inherited from the **network-instance protocols ldp peers** context.

7730 SXR platforms support the following T-LDP hello timers:

- hello holdtime

- hello interval

7730 SXR platforms also support BFD tracking of T-LDP peers for fast failure detection.

5.13.1 Configuring targeted LDP

Procedure

To configure an IPv4 or IPv6 targeted LDP session, use the **network-instance protocols ldp discovery targeted** command.

Example: Configure targeted LDP

The following example shows an IPv4 targeted LDP session with BFD and advertise FEC enabled, custom hello holdtime and interval values defined, and the LSR ID set to the IPv4 address for subinterface ethernet-1/10.1.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols ldp discovery targeted
    network-instance default {
        protocols {
            ldp {
                discovery {
                    targeted {
                        ipv4 {
                            target 10.0.0.1 {
                                admin-state enable
                                enable-bfd true
                                advertise-fec true
                                hello-holdtime 30
                                hello-interval 10
                                override-lsr-id {
                                    subinterface-ipv4 ethernet-1/10.1
                                }
                            }
                        }
                    }
                }
            }
```

```
                }
            }
        }
```

5.14 Manual LDP RLFA

LDP remote LFA (RLFA) builds on the existing segment routing (SR) RLFA capability to compute repair paths to a remote LFA node (or PQ node). After a link failure, RLFA puts packets back onto the shortest path without looping them back to the node that forwarded them. See the *SR Linux Segment Routing Guide* for more information about RLFA computation.

Unlike automatic LDP RLFA, manual LDP RLFA requires the targeted sessions (between source node and PQ node) to be manually configured before a link failure. RLFA repair tunnels can be LDP tunnels only.

Supported platforms

Manual LDP RLFA is supported on 7730 SXR-series platforms only.

General principles of LDP RLFA

The following figure shows the general principles of LDP RLFA operation using an LDP-in-LDP repair tunnel.

Figure 4: General principles of LDP RLFA operation
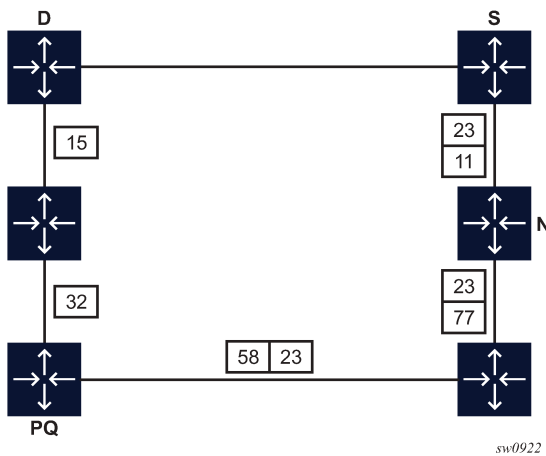
In this figure, S is the source node and D is the destination node. The primary path is the direct link between nodes S and D, and the RLFA algorithm on node S determines the PQ node for destination D. In the event of a failure between nodes S and D, to prevent traffic from looping back to node S, the traffic must be sent to the PQ node. A targeted LDP (T-LDP) session is required between nodes PQ and S. Over that T-LDP session, the PQ node advertises label 23 for FEC D. All other labels are link LDP bindings, which allow traffic to reach the PQ node. On node S, LDP creates an NHLFE that has two labels, where label 23 is the inner label. Label 23 is tunneled to the PQ node, which then forwards traffic on the shortest path to node D.

As prerequisites for RLFA to operate, the LFA algorithm must be run (**loopfree-alternate remote-lfa**), and the PQ node information must also be attached to LDP route entries (**loopfree-alternate augment-route-table**).

Feature considerations

Be aware of the following feature considerations:

- LDP RLFA applies to IPv4 FECs with IS-IS only.

- Manual LDP RLFA requires the targeted sessions (between source node and PQ node) to be manually configured before a link failure. The system does not automatically set up T-LDP sessions toward the PQ nodes that the RLFA algorithm has identified. These targeted sessions must be set up with router IDs that match the IDs the RLFA algorithm uses.

- LDP RLFA is designed to operate in LDP-only environments.

- The **lsp-trace** OAM command is not supported over the repair tunnels.

5.14.1 Configuring manual LDP RLFA

Prerequisites

In IS-IS:

- Enable remote LFA computation using the following command:

  network-instance protocols isis instance loopfree-alternate remote-lfa admin-state enable

- Enable attaching RLFA information to RTM entries using the following command:

  network-instance protocols isis instance loopfree-alternate augment-route-table

  This second command attaches RLFA-specific information to route entries that are necessary for LDP to program repair tunnels toward the PQ node using a specific neighbor.

Procedure

Step 1. To enable manual LDP RLFA on the source node and PQ node, configure targeted LDP sessions using the **network-instance protocols ldp discovery targeted target** command.

Step 2. On the PQ node, enable FEC advertisement using the **network-instance protocols ldp discovery targeted [ipv4 | ipv6] target advertise-fec** command.

Example: Configure manual LDP RLFA on a source node

The following example configures manual LDP RLFA on a source node, with no **advertise-fec** configuration.

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols ldp discovery targeted
    network-instance default {
        protocols {
            ldp {
                discovery {
                    targeted {
                        ipv4 {
                            target 10.1.1.10 {
                                admin-state enable
                                hello-holdtime 60
                                hello-interval 30
                            }
                        }
                    }
                }
            }
        }
    }
```

```
            }
```

Example: Configure manual LDP RLFA on a PQ node

The following example configures manual LDP RLFA on a PQ node, with **advertise-fec** set to **true**.

```
--{ candidate shared default }--[   ]--
# info network-instance default protocols ldp discovery targeted
    network-instance default {
        protocols {
            ldp {
                discovery {
                    targeted {
                        ipv4 {
                            target 10.1.1.15 {
                                admin-state enable
                                advertise-fec true
                                hello-holdtime 60
                                hello-interval 30
                            }
                        }
                    }
                }
            }
        }
    }
```

5.15 Automatic LDP RLFA

Unlike manual LDP RLFA, automatic LDP RLFA automatically establishes targeted LDP sessions without the need to specify a list of peers that require targeted sessions. On the source node, specifying which peer serves as the PQ node is not required. Instead, with automatic RLFA, the source node automatically identifies and establishes targeted LDP sessions with the PQ node.The PQ node then provides an LDP-in-LDP repair path to the destination following a failure. The automatic LDP RLFA feature minimizes overall configuration, providing a more dynamic and flexible environment.

Supported platforms

Automatic LDP RLFA is supported on 7730 SXR-series platforms only.

Simple scenario: one PQ node and one S node

The basic principles of operation for the automatic LDP RLFA capability are the same as in manual LDP RLFA. The following figure again shows the general principles of LDP RLFA operation.

Figure 5: General principles of LDP RLFA operation
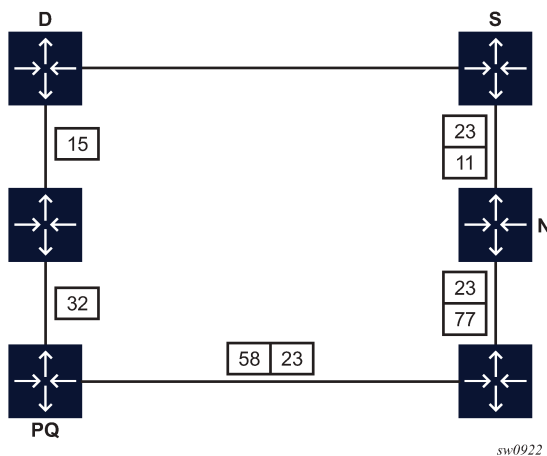
In this example, to address a failure on the shortest path between nodes S and D, node S needs a targeted LDP session toward the PQ node to learn the label-binding information on the PQ node for FEC D. To enable automatic RLFA, the same LFA algorithm used for manual RLFA must be run (**loopfree-alternate remote-lfa**), and the PQ node information must also be attached to LDP route entries (**loopfree-alternate augment-route-table**).

Because node S is the node that requires the T-LDP session, it is configured as the transmitting node (**auto-tx**) to initiate the T-LDP session. The PQ node is configured as the receiving node (**auto-rx**) of the targeted hello request for that session. And similar to manual LDP RLFA, the PQ node requires FEC advertisement to be enabled (**auto-rx advertise-fec enable**) so that the PQ node can send to the S node the label it has bound to FEC D (label 23).

With automatic RLFA enabled, node S uses the PQ node information attached to the route entries to automatically start sending targeted hellos to the PQ node. The PQ node accepts those hellos and the T-LDP session is established.

If the network topology changes and the PQ node of S for FEC D changes, S automatically brings down the T-LDP session to the previous PQ node and attempts to establish a new one towards the new PQ node.

Typical scenario: nodes are both PQ node and S node

In the previous simple example of one S node and one PQ node, **auto-rx** and FEC advertisement can remain disabled on the S node. However, in typical network deployments, each node is potentially the source node and also the PQ node for a given destination FEC. Therefore, all nodes can have both **auto-tx** and **auto-rx** enabled. These nodes can also have other configurations defined for specific LDP peers. These multiple configurations (either explicit or implicit) have implications for the operation of LDP RLFA.

The first implication relates to the fact that LDP operates using precedence levels. When a targeted session is established with a peer, LDP uses the session parameters with the highest precedence. The order of precedence is as follows (in descending order of priority):

1. **peer**
2. **auto-tx**
3. **auto-rx**

To illustrate how LDP precedence affects LDP RLFA, consider a T-LDP session between the following nodes:

- node A (source node)

- node B (PQ node)

If source node A has both a per-peer configuration for node B and **auto-tx** enabled, node A chooses the parameters defined in the per-peer configuration rather than those defined under **auto-tx** to establish the session. A similar precedence also applies on PQ node B. However, when node B uses the per-peer configuration it has for node A rather than the **auto-rx** options (including **advertise-fec enable**), LDP RLFA cannot operate unless the per-peer configuration also enables **advertise-fec**.

A similar logic also applies in that the **auto-tx** configuration takes precedence over **auto-rx**. In a typical scenario, both these modes are enabled on a node that needs to act as both source and PQ node. However, if the node uses the **auto-tx** configuration for the T-LDP session (because it has the highest precedence), **advertise-fec** must be also enabled under **auto-tx** for LDP RLFA to function.

The second implication of multiple LDP configurations is that redundant T-LDP sessions may remain up after a topology change when they are no longer required. In this case, a **tools** command allows you to delete these redundant T-LDP sessions.

Configuration considerations

The following configuration considerations apply:

- automatic LDP RLFA works with IS-IS only

- automatic LDP RLFA only supports IPv4 FECs

- **override-lsr-id** configuration is not supported

- **lsp-trace** on the backup path is not supported

5.15.1 Configuring automatic LDP RLFA

Prerequisites

In IS-IS:

- Enable remote LFA computation using the following command:

  network-instance protocols isis instance loopfree-alternate remote-lfa admin-state enable

- Enable attaching RLFA information to RTM entries using the following command:

  network-instance protocols isis instance loopfree-alternate augment-route-table

Procedure

To enable automatic LDP RLFA, use the **auto-rx** and **auto-tx** options under **network-instance default protocols ldp discovery targeted**, as follows:

- On all potential source nodes, enable **auto-rx**.

- On all potential PQ nodes, enable **auto-tx**.

- On PQ-only nodes, enable the **advertise-fec** option under **auto-rx**.

- On PQ nodes that can also operate as source nodes, enable the **advertise-fec** option under both **auto-rx** and **auto-tx**.

Example: Configure automatic LDP RLFA on node that is both source and PQ node

The following example configures automatic LDP RLFA on a node that is both source and PQ node, with **advertise-fec** set to **true** under both **auto-rx** and **auto-tx**.

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols ldp discovery targeted ipv4
    network-instance default {
        protocols {
            ldp {
                discovery {
                    targeted {
                        ipv4 {
                            auto-rx {
                                admin-state enable
                                advertise-fec true
                            }
                            auto-tx {
                                admin-state enable
                                advertise-fec true
                            }
                        }
                    }
                }
            }
        }
    }
```

5.15.2 Displaying source context for automatic T-LDP session parameters

About this task

It is not possible to configure LDP options specifically for automatic T-LDP sessions except for **advertise-fec**. The session inherits the settings defined for the IPv4 and IPv6 families (under **ldp discovery targeted**) or the system default values. This applies to the following configurations:

- **network-instance default protocols ldp discovery targeted hello-holdtime**
- **network-instance default protocols ldp discovery targeted hello-interval**

Procedure

To identify the context from which the active T-LDP session configuration options are taken, use the **info from state** command.

Example: Display source context for automatic T-LDP session parameters

```
--{ candidate shared default }--[  ]--
# info from state network-instance default protocols ldp discovery targeted ipv4 target
 10.1.1.6 oper-type
    network-instance base {
        protocols {
            ldp {
                discovery {
                    targeted {
                        ipv4 {
                            target 10.1.1.6 {
                                oper-type auto-tx
                            }
                        }
```

```
                    }
                  }
                }
              }
            }
```

5.15.3 Deleting redundant T-LDP sessions

Procedure

To delete any redundant T-LDP sessions, use the **tools network-instance default protocols ldp targeted-auto-rx hold-time** command.

You must run the command during a specific time window on all nodes that have **auto-rx** enabled. The **hold-time** value must be greater than the hello-timer value plus the time required to run the **tools** command on all applicable nodes. Ensure that the configured value is long enough to meet these requirements. A system check verifies that a non-zero value is configured, but no other checks are enforced.

> **Note:** The **tools network-instance default protocols ldp targeted-auto-rx hold-time** command is not synchronized to the standby CPM. If you perform the operation with a large hold-time value and the CPM does a switchover during this time, you must restart the operation on the newly active CPM.

Example: Deleting redundant T-LDP sessions

The following example enables the **targeted-auto-rx hold-time** command to delete any redundant T-LDP sessions.

```
--{ candidate shared default }--[  ]--
# tools network-instance default protocols ldp targeted-auto-rx hold-time 120
```

Example: Display remaining timeout value

While the **targeted-auto-rx hold-time** command is in progress, the following example shows how to display the remaining timeout value using the **info from state** command.

```
--{ candidate shared default }--[  ]--
# info from state network-instance default protocols ldp discovery targeted ipv4 target
  10.10.10.15 hello-adjacencies adjacency 10.10.10.10 label-space-id 10 hello-holdtime
  remaining
```

5.16 LSP ping and trace for LDP tunnels

To check connectivity and trace the path to any midpoint or endpoint of an LDP tunnel, SR Linux supports the following OAM commands:

* **tools oam lsp-ping ldp fec** *<prefix>*

* **tools oam lsp-trace ldp fec** *<prefix>*

Supported parameters include **destination-ip**, **source-ip**, **timeout**, **ecmp-next-hop-select**, and **traffic-class**. However, the only mandatory parameter is **fec**.

Results from the lsp-ping and lsp-trace operations are displayed using **info from state** commands.

For more information, see the *SR Linux OAM and Diagnostics Guide*.

Customer document and product support

Customer documentation
Customer documentation welcome page

Technical support
Product support portal

Documentation feedback
Customer documentation feedback