Nokia Service Router Linux

Release 24.7

MPLS Guide

3HE 20662 AAAA TQZZA
Edition: 01
July 2024

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

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
LDP IPv6	LDP IPv6 Configuring LDP
LSR ID override	Overriding the LSR ID on an interface
Static FEC (FEC originate)	Configuring static FEC (FEC originate)
LDP FEC import and export policies (global and per-peer)	LDP FEC import and export policies

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.

MPLS is currently supported on 7250 IXR platforms.

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.

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.

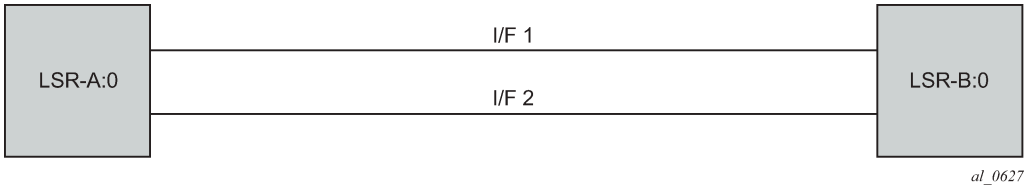
LDP IPv6 is supported for both link and targeted LDP.

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
- 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 FECs
- Automatic FEC origination of the system0.0 /32 IPv4 address prefix
- /32 IPv4 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 prefix FECs into TTM
- Per-peer configurable FEC limit
- Graceful restart helper capability
- BGP shortcuts for IPv4 traffic
- Protocol debug/trace-options
- LDP-IGP synchronization

- Advertise address mapping message for primary IPv4 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
- Show reports for MPLS tunnel tables

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 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   |
+-------------+-------+-------------+-----------+-----+--------+------+----------------+----------+
```

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
- 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 l-LDP: local subinterface IPv4 address
- IPv6 l-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 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:

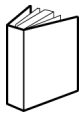
- **tools oam lsp-ping ldp fec** <*prefix*>
- **tools oam lsp-trace ldp fec** <*prefix*>

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