Nokia Service Router Linux
7250 Interconnect Router
7730 Service Interconnect Router

Release 26.3

Segment Routing Guide

3HE 22258 AAAA TQZZA
Edition: 01
March 2026

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 document describes configuration details for the Segment Routing feature set used with the Nokia Service Router Linux (SR Linux).

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how the router is configured.

Note:
This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Software Release Notes* for information on 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

The 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.
- 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.

The following table outlines platform grouping conventions used in the SR Linux documentation suite.

Note: Some platforms in the 7250 IXR support mixed systems. For more information about mixed system support, see "Chassis types" in the *Configuration Basics Guide*.

Table 1: Platform grouping legend

Platform group	Description
7215 IXS	7215 IXS-A1
7220 IXR	All 7220 IXR platforms
7220 IXR-D*x*	7220 IXR-D1, 7220 IXR-D2, 7220 IXR-D2L, 7220 IXR-D3, 7220 IXR-D3L, 7220 IXR-D4, 7220 IXR-D5
7220 IXR-H*x*	7220 IXR-H2, 7220 IXR-H3, 7220 IXR-H4, 7220 IXR-H4-32D, 7220 IXR-H5-32D, 7220 IXR-H5-64D, 7220 IXR-H5-64O
7250 IXR[1]	7250 IXR platforms
7250 IXR Gen 2	7250 IXR-6, 7250 IXR-10
7250 IXR Gen 2c+	7250 IXR-6e with IMM2, 7250 IXR-10e with IMM2, 7250 IXR-X1b, 7250 IXR-X3b
7250 IXR Gen 3	7250 IXR-6e with IMM3, 7250 IXR-10e with IMM3, 7250 IXR-18e, 7250 IXR-X4
7250 IXR-6e/10e (mixed system)	7250 IXR-6e (mixed system)[2], 7250 IXR-10e (mixed system)[2]
7730 SXR	7730 SXR-1-32D, 7730 SXR-1d-32D, 7730 SXR-1x-44S

1 References to the 7250 IXR platform group may be appended with (including mixed systems) or (excluding mixed systems) to indicate mixed system support.

2 References to this platform as part of 7250 IXR (mixed system) indicate mixed system support of 7250 IXR Gen 2c+ (IMM2) and 7250 IXR Gen 3 (IMM3). That is, the 7250 IXR-6e and 7250 IXR-10e can hold and support both IMM2 and IMM3 at the same time.

2 What's new

This section lists the changes that were made in this release.

Table 2: What's new in Release 26.3.1

Topic	Location
Segment Routing traffic statistics	Segment Routing traffic statistics
Supports 7250 IXR Gen 3	• Uncolored SR-MPLS TE-Policy • Colored TE policy • Non-Stop Routing (NSR) for TE-Policy (uncolored or colored) • LSP statistics (Uncolored TE-policy) • LSP statistics (Colored TE policy)
Segment routing with IPv6 data plane (SRv6)	Segment routing with IPv6 data plane (SRv6)
Flexible algorithm (Flex-Algo) for IS-IS	Flexible algorithm (Flex-Algo) for IS-IS
Delay normalization with Flex-Algo	Delay normalization with Flex-Algo

3 About segment routing

Segment routing performs shortest path routing or source routing using the concept of abstract segment. A segment can represent the local prefix of a node, a specific adjacency of the node (interface or next hop), a service context, or a specific explicit path over the network. Each segment is identified by a segment ID (SID).

With segment routing, the source router can define the end-to-end path for packets to reach a destination using an ordered list of segments represented by their SIDs. Each SID represents actions that subsequent nodes in the network execute, such as forwarding a packet to a specific destination or interface.

Unlike a typical Multiprotocol Label Switching (MPLS) architecture, a segment routing network does not require LDP or RSVP-TE to set up tunnels. Instead, segment routing extensions to the IGP (such as IS-IS) provide support for allocation and signaling of SID information.

Supported platforms

Segment routing is supported on the following platforms:

- 7250 IXR
- 7730 SXR

SR-MPLS

When segment routing is instantiated over the MPLS data plane (referred to as SR-MPLS), a SID is represented by a standard MPLS label or an index that maps to an MPLS label. To route a packet to a destination, the source router pushes one or more MPLS labels, which represent the required SIDs onto the packet. Each subsequent node forwards the packet based on the outer label, removes the outer label, and forwards the packet to the next segment on the path.

The MPLS data plane requires no modifications to support SR-MPLS.

SR Linux supports SR-MPLS over both IPv4 (SR-MPLS IPv4) and IPv6 (SR-MPLS over IPv6).

Node SID

A prefix SID is a segment type that represents the ECMP-aware shortest path to reach a particular IP prefix from any IGP topology location. A node SID is a type of prefix SID that identifies a specific node in the IGP topology using a loopback address as the prefix. When a node SID is included in an MPLS label stack, it instructs the receiving node to forward the packet to the destination along the ECMP-aware shortest path determined by the IGP.

You allocate the node SIDs to use in the network in a similar fashion as the loopback IP addresses. Like all prefix SIDs, node SIDs must be unique within the domain, and are allocated in the form of an MPLS label (or index). The range of labels available for prefix and node SIDs is defined in the SRGB.

Protocol-independent prefix SIDs

You can set a prefix SID at the network instance level that is shared by multiple IGPs (currently only IS-IS).

SR Linux supports up to four protocol-independent prefix SIDs to be associated with the default network instance. By default, all prefix SIDs are set as node SIDs. However, you can disable the node SID flag as required.

Note: As an alternative, you can also define a node SID under the IS-IS configuration context. In that case, the node SID is not protocol-independent but is specific to that IGP. However, in that case, the node SID flag cannot be disabled.

You can configure an IS-IS node SID and a protocol-independent prefix SID on a single interface. In this case, the IS-IS IGP overrides the protocol-independent prefix SID configuration, and only the IGP node SID is advertised.

A protocol-independent prefix SID provides more flexiblity over an IS-IS node SID. If required, you can override the protocol-independent prefix SID with an IGP node SID.

SRGB

The Segment Routing Global Block (SRGB) defines the range of MPLS labels available for global segments, such as prefix and node SIDs. The SRGB is defined as one contiguous block of MPLS labels. Nokia strongly recommends using identical SRGBs on all nodes within the SR domain. The use of identical SRGBs simplifies troubleshooting because the same MPLS label represents the same prefix or node SID on each node.

The SRGB configuration is supported on the default network instance only.

SRLB

While the SRGB defines a range of MPLS labels for global segments, the Segment Routing Label Block (SRLB) defines a range of MPLS labels for local segments, such as adjacency SIDs. An adjacency SID is a segment type that represents an instruction to forward packets over a specific (one-hop) adjacency between two nodes in the IGP.

The SRLB is configurable per IGP protocol instance (on the default network instance only). It is defined as one contiguous block of MPLS labels per IGP instance.

SR Linux can support a dynamic SRLB, a static SRLB, or a combination of both.

With a dynamic SRLB, SR Linux dynamically assigns adjacency SIDs as required. With a static SRLB, you must define the static adjacency SID values manually.

Note: Static adjacency SID configuration is not supported for broadcast interfaces.

See the *SR Linux MPLS Guide* for more information about MPLS and MPLS labels.

3.1 IS-IS extensions for segment routing

Segment routing with IS-IS (also known as SR-ISIS) refers to the segment routing extensions of the IS-IS IGP protocol and the forwarding entries created by those extensions. The SR-ISIS extensions advertise TVLs specific to segment routing, which propagate SIDs across the domain.

With segment routing enabled on the IS-IS instance, IS-IS extensions can support TLVs to advertise IPv4/IPv6 prefix SIDs. The following sub-TLVs are defined in RFC 8667 and are supported in the implementation of SR-ISIS:

- Prefix SID sub-TLV
- Adjacency SID sub-TLV
- SR-Capabilities sub-TLV

- SR-Algorithm sub-TLV

By default, SR Linux advertises each configured prefix SID as a node SID. You can disable the node SID flag for protocol-independent prefix SIDs but not for IS-IS-specific node SIDs.

The following sections describes the behaviors and limitations of the SR-ISIS TLV and sub-TLVs.

3.1.1 Router Capability TLV advertisement

In SR Linux, advertisement of the Router Capability TLV (TLV 242) with the sub-TLVs related to segment routing is automatically enabled when segment routing is configured on the IS-IS instance. The originated TLV 242 encodes the router ID and always indicates domain-wide scope. There is no explicit configuration to enable or disable router capability advertisement.

The following table describes the TLV 242 sub-TLVs that SR Linux supports.

Table 3: TLV 242 sub-TLVs

Sub-TLVs	Description
SR Capabilities sub-TLV	Advertises one SRGB block and data plane capability: • If the SRGB block is not configured, the entire SR Capabilities sub-TLV is suppressed • SR Linux does not support multiple, discontinuous SRGB blocks • I = 1 indicates support for IPv4 • V = 1 indicates support for IPv6
SR Algorithm sub-TLV	The algorithm field is set to 0, indicating shortest path first (SPF) algorithm based on link metric. (The value is not checked on receive.)
SR Local Block sub-TLV	(Optional) Advertises the SRLB. SR Linux does not support multiple, discontinuous SRLB blocks.
SRMS Preference sub-TLV	Not supported.

Receiving Router Capability TLVs

SR Linux decodes the received Router Capability TLVs and maintains the details in the LSDB YANG state model.

3.1.2 Prefix SID processing

SR Linux uses Prefix SID sub-TLVs to advertise IPv4/IPv6 prefix SIDs.

Origination of IPv4 and IPv6 Prefix SID sub-TLVs

SR Linux originates Prefix SID sub-TLVs with the following processing rules and flag encoding:

- originates a single Prefix SID sub-TLV per IS-IS IP Reachability TLV
- encodes the 32-bit index in the Prefix SID sub-TLV while the 24-bit label is not supported
- both IPv4 and IPv6 Prefix SID sub-TLVs originate within MT = 0

The following table describes the default flag encoding for Prefix SID sub-TLVs.

Table 4: Default flag encoding for Prefix SID sub-TLV

Flag	Default encoding	Description
R-flag	R = 1 (for node SID)	Re-advertisement flag When R = 1, the Prefix SID sub-TLV and its corresponding IP reachability TLV are propagated between levels.
	R = 0 (for prefix SID)	Re-advertisement flag R = 0 initially, but this setting can change to R = 1 during propagation by other routers.
N-flag	N = 1	Node SID flag Set by default, meaning that SR Linux advertises each configured prefix SID as a node SID. You can disable the node SID flag for protocol-independent prefix SIDs but not for IGP-specific prefix SIDs. If the referenced interface is `system0.0`, the node SID flag must be set.
P-flag	P = 1	No-PHP Flag Always set, meaning the label for the prefix SID is pushed by the PHP router when forwarding to this router. When the SR Linux PHP router (with P-flag set to 1) processes a received prefix SID with the P-flag set to zero, it uses implicit-null for the outgoing label toward the router that advertised it.
E-flag	E = 0	Explicit null flag Always set to 0, meaning that the router is requesting no explicit null termination. However, the SR Linux PHP router processes a received prefix SID with the E-flag set to 1 as long as the P-flag is also set to 1. In this case, the PHP router pushes explicit-null for the outgoing label toward the router that advertised it. The system ignores the value of the E-flag if the P-flag is not set.
V-flag	V = 0	Value flag Always set to 0 to indicate an index for the SID, instead of a specific value.
L-flag	L = 0	Local flag Always set to 0 to indicate that the SID index value is not locally significant.
Algorithm field	algorithm 0	Always set to 0 to indicate shortest path first (SPF) algorithm based on link metric. This field is not checked on a received Prefix SID sub-TLV.

Receiving IPv4 and IPv6 Prefix SID sub-TLVs

SR Linux processes a Prefix SID sub-TLV using the following rules:

- decodes Prefix SID sub-TLVs that are received in TLVs 135, 235, 236, and 237 for representation in the YANG state model of the LSDB
- ignores Prefix SID sub-TLVs received in TLVs 235 and 237 for further processing; these sub-TLVs do not appear in the state representation of the local SID database (only in the LSDB)
- if the local flag is set or the algorithm is not 0, deems a Prefix SID sub-TLV in a received TLV 135 or TLV-236 invalid and makes it inactive
  - If the MPLS label value implied by the received Prefix SID sub-TLV falls outside the range of the SRGB block, deems the node SID invalid and makes it inactive (reason = sid-index-out-of-range). No forwarding state is created for invalid entries.
- does not resolve a Prefix SID sub-TLV received with the N-flag set and a prefix length different than 32 (for IPv4) or 128 (for IPv6)
- does not resolve a Prefix SID sub-TLV received in TLV 135 or TLV 236 if the L-flag is set
- resolves a Prefix SID received within an IP reachability TLV based on the following route preference:
  1. SID received via L1 in a Prefix SID sub-TLV part of the IP reachability TLV
  2. SID received via L2 in a Prefix SID sub-TLV part of the IP reachability TLV
- resolves a Prefix SID sub-TLV received without the N-flag set as long as the prefix length is 32 (for IPv4) or 128 (for IPv6)
- processes only the first Prefix SID sub-TLV if multiple are received within the same IS-IS IP reachability TLV

Inter-level propagation of prefix SIDs

SR Linux performs inter-level propagation of prefix SIDs as follows:

- An L1L2 router propagates a prefix (and Prefix SID sub-TLV) received in an IP reachability TLV from L1 to L2. A router in L2 sets up an SR-ISIS tunnel to the L1 router via the L1L2 router, which acts as an LSR.
- If an ABR summarizes a prefix, the Prefix SID sub-TLV is not propagated with the summarized route between levels. To propagate the node SID for a /32 prefix, you must disable route summarization.
- By default, an L1L2 router does not propagate a prefix (or Prefix SID sub-TLV) received in an IP reachability TLV from L2 to L1.
- Using an export policy, an L1L2 router propagates a prefix (and Prefix SID sub-TLV) received in an IP reachability TLV from L2 to L1. A router in L1 sets up an SR-ISIS tunnel to the L2 router via the L1L2 router, which acts as an LSR.
  - L2 to L1 route leaking occurs when a level 2 IS-IS route is matched by the IS-IS export policy.

IS-IS prefix SID database

In the YANG state model, the IS-IS prefix SID database captures information about:

- advertised IS-IS node SIDs
- remotely originated node and prefix SIDs learned and marked as active by IS-IS

Global SID Database

In the YANG state model, the global SID database captures information about:

- configured protocol-independent prefix SIDs
- advertised IS-IS node SIDs
- remotely originated node and prefix SIDs learned and marked as active by IS-IS

Prefix SID conflicts

Remotely signaled prefix SID entries can cause conflicts with local prefix SIDs in the following cases:

- **prefix conflict**
  If you configure an SID index value for a prefix/node SID that creates an overlap with an existing ILM entry in the SRGB block, a prefix conflict occurs.

  In SR Linux, all local prefix/node SID entries have a lower numerical preference than remote prefix SID entries learned via IS-IS. Therefore the local prefix/node SID remains active and advertised, but in the SID database the entry appears with a status of: `prefix-conflict = true`. All other conflicting entries show as inactive in the IS-IS prefix SID database and may no longer be visible in the global SID database.
- **SID conflict**
  After SR Linux removes the inactive prefix SID entries, an SID conflict can still occur if the same SID is assigned to multiple IS-IS prefixes. In this case, the prefix SID with the lowest SID index remains active, but in the SID database the entry appears with a status of: `sid-conflict = true`. All other conflicting entries show as inactive in the IS-IS prefix SID database and may no longer be visible in the global SID database.
- **SID out of range**
  When a prefix SID advertised from another router has a SID index or label value that is not within the locally defined SRGB range of the network instance, SR Linux determines that it is out of range. All other conflicting entries show as inactive in the IS-IS prefix SID database and may no longer be visible in the global SID database.

3.1.3 Adjacency SID processing

SR Linux uses Adjacency SID sub-TLVs to advertise IPv4/IPv6 adjacency SIDs.

Origination of IPv4 and IPv6 Adjacency SID sub-TLVs

By default, SR Linux does not automatically assign adjacency SIDs, but you can enable dynamic adjacency SID assignment on the IS-IS instance. SR Linux supports origination of IPv4 and IPv6 Adjacency SID sub-TLVs in TLVs 22 and 222. Dynamic adjacency SIDs are allocated as follows:

- SR-ISIS assigns dynamic adjacency SIDs on all P2P and LAN IS-IS interfaces in all levels (except for interfaces individually configured with an adjacency SID assignment of **none** or **static**).
- If an interface is enabled for IPv4, SR-ISIS allocates IPv4 adjacency SIDs.
- If the interface is enabled for IPv6, SR-ISIS allocates IPv6 adjacency SIDs.
- For each IS-IS interface, the YANG state indicates all IPv4 and IPv6 adjacency SIDs that are currently active and programmed in the ILM table.

The following table describes the flag encoding for the Adjacency SID sub-TLVs.

Table 5: Flag encoding for Adjacency SID sub-TLV

Flag	Encoding	Description
F-flag	F = 0 (for IPv4)	Address-family flag for IPv4 adjacency encapsulation.
	F = 1 (for IPv6)	Address-family flag for IPv6 adjacency encapsulation.
B-flag	B = 0	Backup flag. Set to zero and is not processed on receipt.
V-flag	V = 1	Value flag. Always set to 1, indicating that the adjacency SID carries a value.
L-flag	L = 1	Local flag. Always set to 1, indicating that the adjacency SID has local significance.
S-flag	S = 0	Set flag. Always set to zero because assigning adjacency SIDs to parallel links between neighbors is not supported. A received adjacency SID with the S-flag set is not processed.
P-Flag	P = 1 (static)	Set to 1 when static adjacency-sid is configured on the interface, indicating that the adjacency SID allocation is persistent.
	P = 0 (dynamic)	Set to 0 when dynamic adjacency-sid is configured on the interface.
weight octet	N/A	Not supported and is set to all zeros.

Receiving IPv4 and IPv6 Adjacency SID sub-TLVs

- LAN Adjacency SID sub-TLVs received in TLVs 22 and 222 are decoded for representation in the YANG state model of the LSDB. If there are multiple Adjacency SID sub-TLVs in a particular TLV, only the first Adjacency SID sub-TLV is processed.
- LAN Adjacency SID sub-TLVs received in TLVs 23 and 223 are ignored for further processing.
- A LAN Adjacency SID sub-TLV in a received TLV 22/222 is invalid and not used if either of the following are true:
  - The V-flag is not set.
  - The L-flag is not set.
- A received adjacency SID with the S-flag set is not processed.

3.1.4 Datapath programming by SID type

The following table describes the datapath programming by SID type.

Table 6: Datapath programming by SID type

SID type	Datapath programming
IS-IS node SID	When advertised, the IS-IS node SID creates an ILM entry that matches the associated MPLS label value (SRGB start-label + index) and performs a POP operation. The POP operation indicates that the default network instance is the context for the IP header lookup that follows. **Note:** On a 7250 IXR Gen 3, a transit router (LSR) forwards labeled packets based on the segment routing information distributed via IS-IS extensions. For details, see the "Transit routers on 7250 IXR Gen 3" section of the SR Linux MPLS Guide.
Protocol-independent prefix SID	When configured, the protocol-independent prefix SID (whether advertised or not), creates an ILM entry that matches the associated MPLS label value (SRGB start-label + index) and performs a POP operation. The POP operation indicates that the default network instance is the context for the IP header lookup that follows.
IS-IS adjacency SID (P2P and LAN)	When advertised, the adjacency SID: • Creates an ILM entry that matches the associated MPLS label value and forwards matching packets to the adjacent neighbor using a SWAP to implicit null. – If the adjacency SID is IPv4, forwarding is via an IPv4 ARP entry. – If the adjacency SID is IPv6, forwarding is via an IPv6 ND entry. • Creates a TTM entry with SR-ISIS as the tunnel type. The destination of the tunnel is the IPv4/IPv6 interface address of the neighbor.
Remote prefix SID	When received and valid: • The remote prefix SID: – Creates an ILM entry that matches on the localized MPLS label value (SRGB start-label + index) and performs a SWAP operation. – Creates an IPv4/IPv6 tunnel entry in TTM with SR-ISIS as the tunnel type. • If IS-IS has a non-ECMP route toward the IPv4/IPv6 prefix, then the FEC linked to the ILM (or representing the tunnel) has one next-hop label forwarding entry (NHLFE). The NHLFE pushes label N, where N = (SRGB start label of the next-hop IS-IS neighbor) + SID index. • If IS-IS has an ECMP route toward the IPv4/IPv6 prefix, then the FEC linked to the ILM (or representing the tunnel) has multiple ECMP members, each corresponding to one NHLFE. Each NHLFE pushes label Ni, where Ni = (SRGB start label of next-hop IS-IS neighbor) + SID index.

3.1.5 Segment Routing traffic statistics

Note: This feature is currently supported exclusively on the 7730 SXR platform for SR-ISIS.

When implementing traffic engineering in an SR-enabled network, it is essential to have visibility into the traffic volumes on a per-SID basis. Segment Routing traffic statistics are often used in SDN applications, via Path Computation Engine (PCE) to perform better load balancing of traffic across various paths throughout the network.

3.1.5.1 Enabling statistics collection for SR-ISIS

Procedure

SR Linux supports enabling ingress and egress statistics collection for both adjacency SIDs and node SIDs within the IS-IS segment routing context (`network-instance protocols isis instance segment-routing.mpls`).

Example: Enabling ingress statistics collection for SR-ISIS SIDs

In the following configuration example, the ingress statistics collection is enabled for both adjacency SIDs and node SIDs on an IS-IS instance *sr-isis-1*.

```
A:root@srl1# /info with-context network-instance default protocols isis instance sr-isis-1
 segment-routing mpls ingress-statistics
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    segment-routing {
                        mpls {
                            ingress-statistics {
                                adj-sid true
                                node-sid true
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Enabling egress statistics collection for SR-ISIS SIDs

In the following configuration example, the egress statistics collection is enabled for both adjacency SIDs and node SIDs on an IS-IS instance *sr-isis-2*.

```
A:root@srl1# /info with-context network-instance default protocols isis instance sr-isis-2
 segment-routing mpls egress-statistics
    network-instance default {
        protocols {
            isis {
                instance sr-isis-2 {
                    segment-routing {
                        mpls {
```

```
                        egress-statistics {
                            adj-sid true
                            node-sid true
                        }
                    }
                }
            }
        }
    }
}
```

3.2 Supported functionality

SR-MPLS over IPv4 and SR-MPLS over the IPv6 dataplane

- maximum of one label pushed per packet (corresponding to a remote node SID)
- maximum of one label popped per packet (corresponding to a local node SID)
- transit and egress LSR ECMP hashing
- ingress LSR ECMP hashing
- prefix/node SID (IPv4/IPv6)
- adjacency SID (IPv4/IPv6)

MPLS label management

- Configurable SRGB for the default network instance:
  - SRGB is a reference to a static shared label range.
  - SRGB must be one contiguous block.
- Configurable SRLB for the IS-IS instance of the default network instance:
  - SR Linux supports a static SRLB (dedicated or shared), a dynamic SRLB (dedicated only), or a combination of both.
  - Each SRLB must be one contiguous block.

IS-IS segment routing extensions

- IPv4/IPv6 prefix/node SID
- SID collision handling (no event generated when collision detected)

BGP shortcuts over segment routing tunnels

- SR-ISIS-IPv4 tunnels in TTM can resolve BGP IPv4 routes.
- SR-ISIS-IPv6 tunnels in TTM can resolve:
  - BGP IPv4 routes
  - BGP IPv6 routes
- SR-ISIS tunnels are programmed into TTM with a non-configurable preference value of 11.
- SR-ISIS tunnels that correspond to an adjacency-SID have a metric of 0.

- SR-ISIS tunnels that correspond to a remote prefix-SID have a metric that reflects the IGP cost to reach the advertising router.

SR-MPLS OAM

- ICMP tunneling
- RFC 4950 extensions

ICMP tunneling

SR Linux supports ICMP extensions for MPLS to support debugging and tracing in MPLS and SR-MPLS networks. With ICMP tunneling enabled, ICMP messages can be tunneled to the endpoint of the tunnel and then returned through IP routing. For more information, see the *SR Linux MPLS Guide*.

4 SR-MPLS configuration on the default network instance

Segment routing on the MPLS data plane is supported on the default network instance only. To configure available SR-MPLS options for the default network instance, perform the following tasks:

1. Defining the SRGB and enabling SR-MPLS
2. Defining protocol-independent prefix SIDs (optional)

4.1 Defining the SRGB and enabling SR-MPLS

About this task

The SRGB references a static shared MPLS label range that must be one contiguous block of labels. To configure the SRGB, you must enter the **segment-routing mpls** context on the default network instance, which enables SR-MPLS.

Note: Nokia strongly recommends to using identical SRGBs on all nodes within the SR domain.

Procedure

Step 1. Define a static MPLS label range in the system context.

Step 2. Assign that MPLS label range to be available for use by the SRGB.

Example: Define the MPLS label range for the SRGB

The following example defines static shared MPLS label range `srgb-range-1` for the SRGB.

```
--{ * candidate shared default }--[ ]--
# info with-context system mpls label-ranges static srgb-range-1
    system {
        mpls {
            label-ranges {
                static srgb-range-1 {
                    shared true
                    start-label 16001
                    end-label 16999
                }
            }
        }
    }
```

Example: Assign the MPLS label range to the SRGB

The following example assigns static label range `srgb-range-1` to the SRGB.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default segment-routing mpls global-block
    network-instance default {
        segment-routing {
```

```
            mpls {
                global-block {
                    label-range srgb-range-1
                }
            }
        }
    }
```

4.2 Defining protocol-independent prefix SIDs

About this task

You can configure a prefix SID that is shared by multiple IGPs (currently only IS-IS) in a network instance. SR Linux supports up to four protocol-independent prefix SIDs to be associated with the default network instance. By default, all prefix SIDs are set as node SIDs. However, you can disable the node SID flag as required.

Note: As an alternative, you can define a node SID under the IS-IS configuration context. In that case, the node SID is not protocol-independent but is specific to that IGP, and the node SID flag cannot be disabled.

A single interface can be configured with both an IS-IS node SID and a protocol-independent prefix SID. In this case, the IS-IS IGP overrides the protocol independent prefix SID configuration, and only the IGP node SID is advertised.

A protocol-independent prefix SID is typically preferred over an IS-IS node SID because the protocol-independent prefix SID provides more flexibility. If required, you can override the protocol-independent prefix SID with an IGP node SID.

Prerequisites

- Configure the primary address of a loopback (loN.n) or system0.0 subinterface with the required node SID prefix.

Procedure

Step 1. To configure the local prefix SID, you must specify the following:

- local-prefix SID index (1 to 4)
- loopback subinterface that owns the advertised prefixes
- IPv4 or IPv6 (or both) label index for the SID, referencing the SRGB base, using one of the following options:
  ipv4-label-index | ipv6-label-index

Step 2. Optionally, you can disable the node SID flag on the defined prefix SID, using the following option:

node-sid {true | false}
If the referenced interface is system0.0, the node SID flag cannot be set to false.

Example: Define local prefix SID

The following example defines local prefix SID 2 on loopback interface lo0.2. Prefix SID 2 is associated with IPv4 label index 102 and IPv6 label index 202, and the node SID flag is set to false.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default segment-routing mpls local-prefix-sid 2
    network-instance default {
        segment-routing {
            mpls {
                local-prefix-sid 2 {
                    interface lo0.2
                    ipv4-label-index 102
                    ipv6-label-index 202
                    node-sid false
                }
            }
        }
    }
```

5 SR-MPLS configuration on the IS-IS instance

SR-MPLS is configurable on one IS-IS instance only (on the default network instance). Before you can enable SR-MPLS you must configure IS-IS as the IGP on the segment routing nodes in your network. See the *SR Linux Routing Protocols Guide*.

SR Linux supports a static or dynamic SRLB, or a combination of both. With dynamic SRLB, SR Linux automatically assigns the dynamic adjacency SIDs for the associated interfaces. With static SRLB, you must manually define static adjacency SIDs for the associated interfaces.

To configure SR-MPLS on an IS-IS instance:

1. Specify the node SID (a protocol-independent prefix SID or IS-IS node SID). See:
   - Defining protocol-independent prefix SIDs
   - Configuring an IS-IS node SID
2. Define the Segment Routing Label Block (SRLB) and configure the adjacency SID mode using one of the following methods:
   - Dynamic adjacency SIDs configuration
   - Static adjacency SIDs configuration
3. (Optional) Set interface-specific adjacency SID modes. See:
   - Overriding adjacency SID assignment mode on an interface
4. Enable SR-MPLS. See:
   - Enabling SR-MPLS on the IS-IS instance

5.1 Configuring an IS-IS node SID

Prerequisites

- Configure the primary address of a loopback (loN.n) or system0.0 subinterface with the prefix to associate with the node SID, and set the subinterface as an IS-IS interface.

About this task

Unlike protocol-independent prefix SIDs, IS-IS node SIDs are specific to the IS-IS IGP. Also, you cannot disable the node SID flag for IS-IS node SIDs.

Note: The IS-IS node SID configuration is optional if you have already defined a protocol-independent prefix SID. If both are applied to the same interface, the IS-IS IGP node SID configuration overrides the protocol-independent prefix SID configuration.

To define the node SID in the IS-IS interface configuration, you must specify an index value that is relative to the SRGB. In this case, the MPLS label value is calculated as follows:

Local Label (Prefix/Node SID) = SRGB start-label + {SID index}

For example, if the SRGB range starts at 16000, and the index value is 1, the resulting MPLS label value for the SID is 16001.

Procedure

To configure the node SID corresponding to an IPv4 or IPv6 prefix, assign an index value to the loopback (or system) subinterface in the IS-IS instance.

Example: Configure node SID

The following example adds IPv4 node SID index 1 to loopback subinterface lo0.1.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance sr-isis-1 interface
 lo0.1
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    interface lo0.1 {
                        segment-routing {
                            mpls {
                                ipv4-node-sid {
                                    index 1
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

5.2 Dynamic adjacency SIDs configuration

To enable allocation of dynamic adjacency SIDs for all interfaces in the IS-IS instance, perform the following tasks:

1. Defining the dynamic SRLB
2. Enabling dynamic adjacency SID assignment for the IS-IS instance

Note: You can optionally override the dynamic adjacency SID setting for the IS-IS instance by configuring individual interfaces to use static adjacency SIDs or to use no adjacency SID assignment at all. See section Overriding adjacency SID assignment mode on an interface for more information.

5.2.1 Defining the dynamic SRLB

About this task

To assign dynamic adjacency SID labels, you must configure a dynamic SRLB that references a dynamic range of MPLS labels. The dynamic SRLB must reference a dedicated (non-shared) label range consisting of one contiguous block of labels.

Procedure

Step 1. Define a dynamic MPLS label range.

Step 2. Assign that MPLS label range to the SRLB.

Example: Define the dynamic MPLS label range for the SRLB

The following example defines a dynamic MPLS label range (`srlb-dynamic-1`) for use by the SRLB.

```
--{ * candidate shared default }--[ ]--
# info with-context system mpls label-ranges dynamic srlb-dynamic-1
    system {
        mpls {
            label-ranges {
                dynamic srlb-dynamic-1 {
                    start-label 15001
                    end-label 15999
                }
            }
        }
    }
```

Example: Assign the dynamic MPLS label range to the SRLB

The following example assigns the defined dynamic MPLS label range (`srlb-dynamic-1`) to the SRLB.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis dynamic-label-block
    network-instance default {
        protocols {
            isis {
                dynamic-label-block srlb-dynamic-1
            }
        }
    }
```

Note: SR Linux advertises SRLB via Router-capability TLV.

5.2.2 Enabling dynamic adjacency SID assignment for the IS-IS instance

About this task

You can enable dynamic adjacency SID allocation for the network instance. The default setting is disabled (**false**). When enabled, IS-IS assigns a dynamic adjacency SID to all IS-IS interfaces in all levels, except for interfaces configured at the interface level with an adjacency SID assignment of **none** or **static**. You can also set the hold time that is applied to the dynamically allocated adjacency SIDs.

Procedure

To enable dynamic adjacency SID assignment, in the IS-IS instance configuration, set the segment routing **dynamic-adjacency-sids** option to **all-interfaces true**.

Example: Enable dynamic adjacency SIDs for all interfaces

The following example enables dynamic adjacency SIDs for IS-IS instance sr-isis-1 and sets the hold time to 20 seconds.

```
--{ * candidate shared default }--[ ]--
```

```
# info with-context network-instance default protocols isis instance sr-isis-1 segment-
routing mpls
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    segment-routing {
                        mpls {
                            adjacency-sid-hold-time 20
                            dynamic-adjacency-sids {
                                all-interfaces true
                            }
                        }
                    }
                }
            }
        }
    }
```

5.3 Static adjacency SIDs configuration

To enable assignment of static adjacency SIDs for all interfaces in the IS-IS instance, perform the following tasks:

1. Defining the static SRLB
2. Configuring static adjacency SID assignment for the IS-IS instance
3. Configuring static adjacency SIDs for an interface

Note:

- Static adjacency SID configuration is not supported for broadcast interfaces.
- You can optionally override the static adjacency SID setting by configuring individual interfaces to use dynamic adjacency SIDs. See Overriding adjacency SID assignment mode on an interface.

5.3.1 Defining the static SRLB

About this task

To assign static adjacency SID labels, you must configure a static SRLB that references a static range of MPLS labels. The static SRLB must reference a dedicated or shared label range consisting of one contiguous block of labels.

Procedure

Step 1. Define a static MPLS label range.

Step 2. Assign that MPLS label range to the static SRLB.

Example: Define the static MPLS label range for the SRLB

The following example defines a static MPLS label range (`srlb-static-10`) for use by the SRLB.

```
--{ * candidate shared default }--[ ]--
# info with-context system mpls label-ranges static srlb-static-10
```

```
    system {
        mpls {
            label-ranges {
                static srlb-static-10 {
                    shared true
                    start-label 14001
                    end-label 14999
                }
            }
        }
    }
```

Example: Assign the static MPLS label range to the SRLB

The following example assigns the defined static MPLS label range (`srlb-static-10`) to the static SRLB label block.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 segment-
routing mpls static-label-block
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    segment-routing {
                        mpls {
                            static-label-block srlb-static-10
                        }
                    }
                }
            }
        }
    }
```

5.3.2 Configuring static adjacency SID assignment for the IS-IS instance

About this task

You can configure the IS-IS instance to use static adjacency SID assignment. In this case, IS-IS does not assigns any dynamic adjacency SIDs to any interfaces.

Procedure

To specify static adjacency SID assignment for all interfaces in the IS-IS instance, set the segment routing **dynamic-adjacency-sids** option to **all-interfaces false**.

Example: Enable static adjacency SIDs for all interfaces

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 segment-
routing mpls dynamic-adjacency-sids
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    segment-routing {
                        mpls {
                            dynamic-adjacency-sids {
                                all-interfaces false
                            }
```

```
                    }
                }
            }
        }
    }
}
```

5.3.3 Configuring static adjacency SIDs for an interface

About this task

If you set the adjacency SID mode to static for the IS-IS instance or for specific interfaces, you must manually set the value for the adjacency SIDs for the applicable interfaces.

Procedure

To define the static adjacency SID value for an interface, use the following options:

- **ipv4-adjacency-sid static** <*adjacency-SID*>
- **ipv6-adjacency-sid static** <*adjacency-SID*>

Example: Set static IPv4 and IPv6 adjacency SID values

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 interface
 ethernet-2/1.1 segment-routing mpls
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    interface ethernet-2/1.1 {
                        segment-routing {
                            mpls {
                                ipv4-adjacency-sid {
                                    static 14001
                                }
                                ipv6-adjacency-sid {
                                    static 14501
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

5.4 Overriding adjacency SID assignment mode on an interface

About this task

You can override the adjacency SID configuration set on the IS-IS instance by configuring individual interfaces with a different adjacency SID assignment mode. The SID adjacency options available for an interface are:

- **dynamic**

IS-IS dynamically allocates one or more dynamic adjacency SIDs for this interface for each enabled address family. On a broadcast interface, the TLV 22/222 corresponding to the adjacency with the Designated IS (DIS) includes a LAN adjacency SID sub-TLV that reports all the adjacent systems on the LAN. In this case, you must also configure a dynamic SRLB to automatically assign the adjacency SIDs to the dynamic interfaces.

- **static**
  IS-IS does not assign dynamic adjacency SIDs. Instead, you must statically configure an adjacency SID for the interface. In this case, you must also configure a static SRLB. This option is not available if the interface type is broadcast (LAN).
- **none**
  No adjacency SIDs are allocated. If no SR-MPLS traffic is flowing on a particular interface, set the adjacency SID assignment to **none** to save resources that the dynamic adjacency SIDs would otherwise consume.

Procedure

Under the IS-IS interface, set the following segment routing options:

- **ipv4-adjacency-sid assignment [static | none | dynamic]**
- **ipv6-adjacency-sid assignment [static | none | dynamic]**

Example: Set IPv4 and IPv6 adjacency SID assignments to static

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 interface
 ethernet-1/2.1
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    interface ethernet-1/2.1 {
                        segment-routing {
                            mpls {
                                ipv4-adjacency-sid {
                                    assignment static
                                }
                                ipv6-adjacency-sid {
                                    assignment static
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Set IPv4 and IPv6 adjacency SID assignment to dynamic

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 interface
 ethernet-1/2.2
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    interface ethernet-1/2.2 {
                        segment-routing {
```

```
                        mpls {
                            ipv4-adjacency-sid {
                                assignment dynamic
                            }
                            ipv6-adjacency-sid {
                                assignment dynamic
                            }
                        }
                    }
                }
            }
        }
    }
```

5.5 Enabling SR-MPLS on the IS-IS instance

Procedure

To enable the IS-IS extensions for advertisement of segment routing capabilities, you must enable SR-MPLS on the IS-IS instance.

Example: Enable SR-MPLS

The following example shows the SR-MPLS configuration on IS-IS instance `sr-isis-1`.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 segment-
routing mpls
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    segment-routing {
                        mpls {
                        }
                    }
                }
            }
        }
    }
```

6 Loop-free alternates

SR Linux supports the following loop-free alternate (LFA) features with SR-ISIS:

- LFA
- Remote LFA (RLFA)
- Topology-independent LFA (TI-LFA)

Note:

- SR Linux does not support multi-homed prefix LFA extensions.
- SR Linux does not support disabling LFA protection for adjacency SIDs.

6.1 LFA

The LFA feature provides backup next hops for IP prefixes if the primary next hop becomes unavailable. When LFA is enabled, the IGP SPF attempts to compute both a primary next hop and an LFA next hop for every learned prefix. The LFA next hop is populated in the routing table along with the primary next hop for the prefix. If a failure occurs, traffic can be rerouted to the precomputed LFA next hop, minimizing packet loss and convergence time.

With LFA operation, the repair node must be directly connected to the point of local repair (PLR), and therefore no tunnels are required. The repair node, in its pre-convergence state, must not route traffic to the destination through the PLR. In addition, the repair node's total cost to the destination must be less than the PLR's total cost to the destination.

6.2 Configuring LFA

Procedure

To enable LFA on an IS-IS instance, use the **loopfree-alternate admin-state enable** command.

Example: Configuring LFA

The following example enables LFA on an IS-IS instance.

```
--{  candidate shared default }--[  ]--
# info with-context network-instance default protocols isis
    network-instance default {
        protocols {
            isis {
                instance name {
                    loopfree-alternate {
                        admin-state enable
                    }
                }
            }
        }
```

```
    }
```

6.3 Remote LFA with segment routing

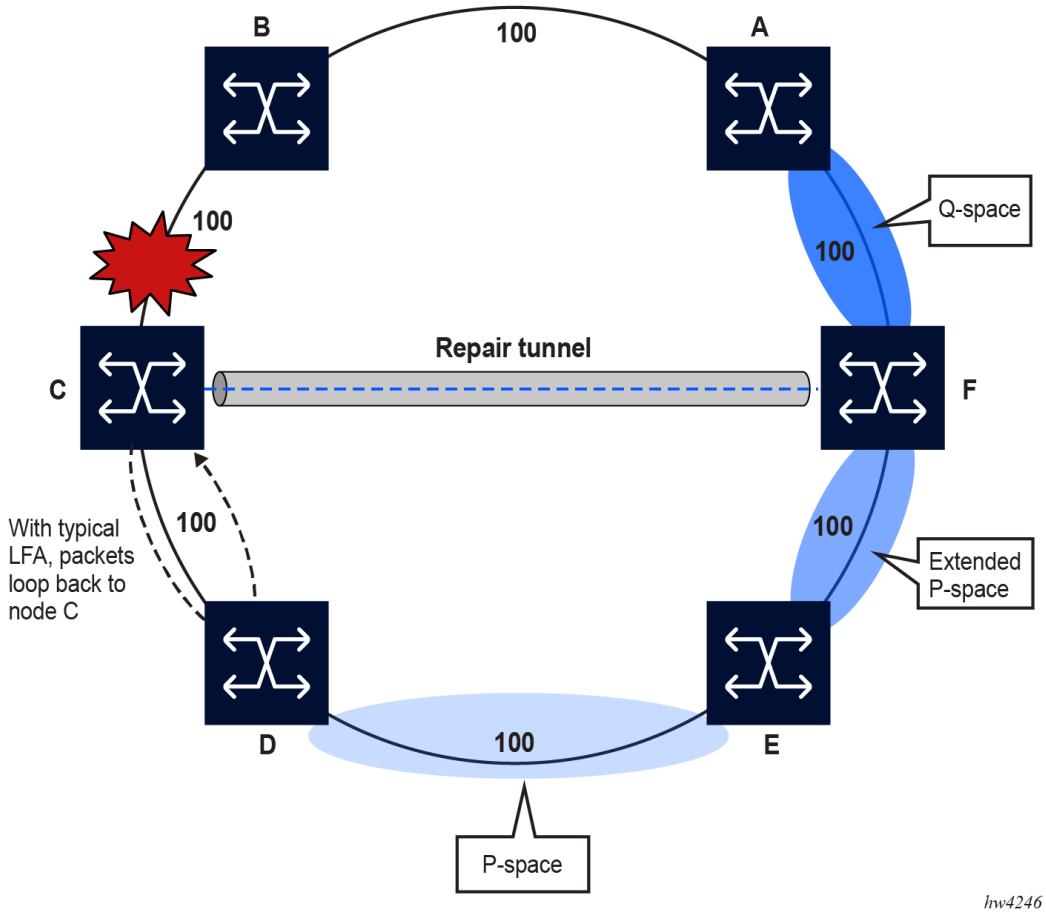
Remote LFA (RLFA) extends the protection coverage of LFA-FRR to any topology by automatically computing and establishing a repair tunnel to a remote LFA node following a link failure. Remote LFA puts the packets back into the shortest path without looping them back to the node that forwarded them.

The remote LFA algorithm for link protection is described in RFC 7490, *Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)*. Unlike a typical LFA calculation, which is calculated per prefix, the LFA algorithm for link protection is a per-link LFA SPF calculation. The algorithm provides protection for all destination prefixes that share the protected link by using the neighbor on the other side of the protected link as a proxy for all of the remote destinations.

6.3.1 Remote LFA PQ node algorithm

The following figure shows an example of RLFA enabled in a ring topology.

Figure 1: Example of remote LFA topology



If the typical LFA SPF in node C computes the per-prefix LFA next hop, prefixes that use link C-B as the primary next hop have no LFA next hop because of the ring topology. For example, if node C uses node link C-D as a backup next hop, node D would loop the packets back to node C.

However, with remote LFA enabled, node C runs the following PQ algorithm, per RFC 7490.

1. **Compute the extended P space of node C with respect to link C-B.**

   The extended P space is the set of nodes that are reachable from node C without any path transiting the protected link (link C-B). This computation yields nodes D, E, and F.

   The determination of the extended P space by node C uses the same computation as the regular LFA by running SPF on behalf of each of the neighbors of C.

   Note: Per RFC 7490, the P space of node C excludes node F because node C has two ECMP paths to node F, one of which traverses link C-B. This exclusion prevents node C from attempting a repair via the failed link C-B. However, with remote LFA enabled, node F is included in the extended P space for node C.

2. **Compute the Q space of node B with respect to link C-B.**

The Q space is the set of nodes that can reach the destination proxy (node B) without any path transiting the protected link (link C-B).

The Q space calculation is effectively a reverse SPF of node B. In general, one reverse SPF is run on behalf of each neighbor of C to protect all destinations resolving over the link to the neighbor. This yields nodes F and A.

3. **Select the best alternate node.**

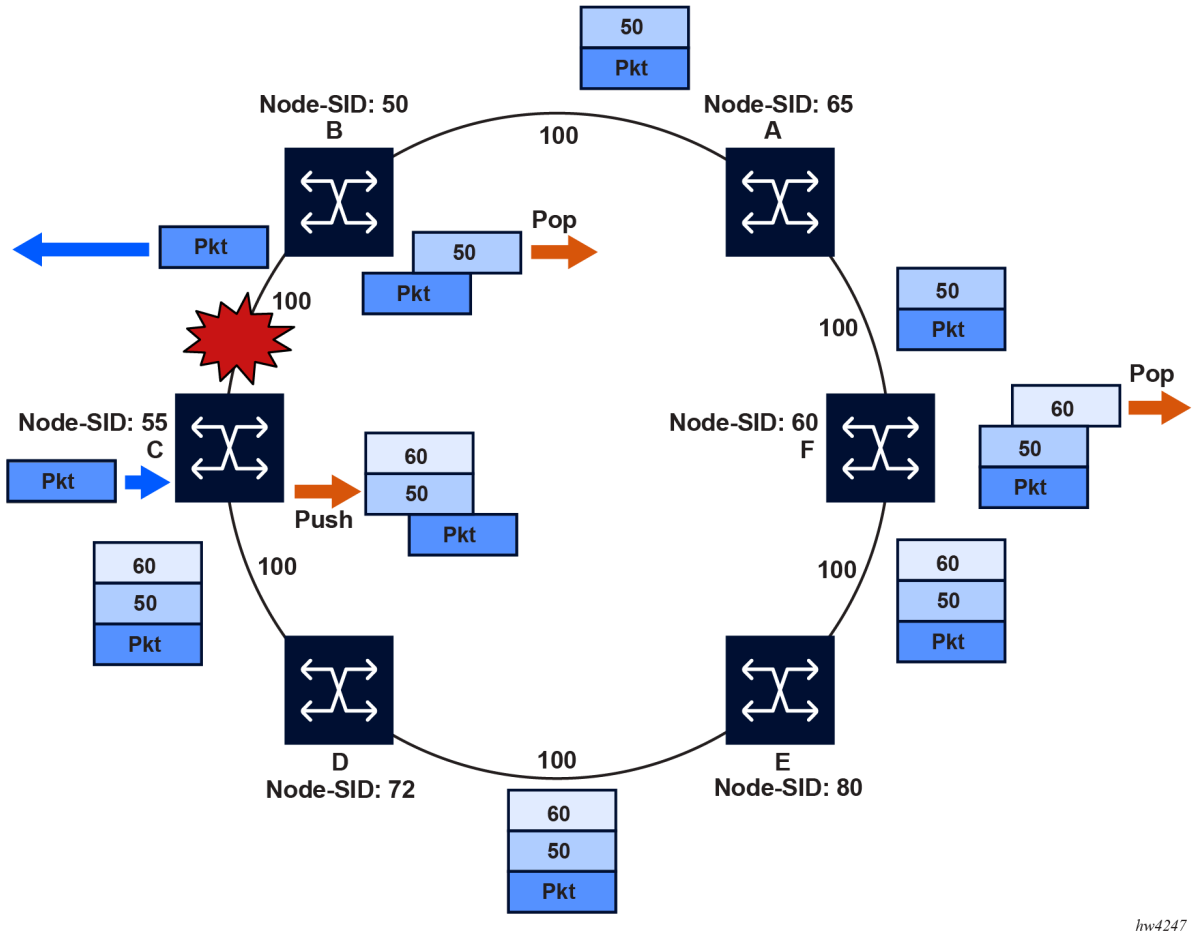
   This is the intersection of extended P and Q spaces. The best alternate node (PQ node) is node F in the preceding figure. From node F onwards, traffic follows the IGP shortest path.

   If many PQ nodes exist, the lowest IGP cost from node C is used to narrow down the selection, and if more than one PQ node remains, the node with the lowest router ID is selected.

6.3.2 Label stack encoding

The details of the label stack encoding when the packet is forwarded over the remote LFA next hop are shown in the following figure.

Figure 2: Remote LFA next-hop in segment routing



The label corresponding to the node SID of the PQ node is pushed on top of the original label of the SID of the resolved destination prefix. If node C has resolved multiple node SIDs corresponding to different prefixes of the selected PQ node, it pushes the lowest node SID label on the packet when forwarded over the remote LFA backup next hop.

If the PQ node is also the advertising router for the resolved prefix, the label stack is compressed. In IS-IS, the label stack is always reduced to a single label, which is the label of the resolved prefix owned by the PQ node.

6.3.3 Remote LFA rules and limitations

The following rules and limitations apply to the remote LFA implementation.

- If you exclude a network IP interface from being used as an LFA next-hop using the **loopfree-alternate exclude** command under the IS-IS context of the interface, the interface is also excluded from being used as the outgoing interface for a remote LFA tunnel next hop.
- As with the regular LFA algorithm, the remote LFA algorithm computes a backup next hop to the ABR advertising an inter-area prefix and not to the destination prefix.

6.3.4 Remote LFA node-protect operation

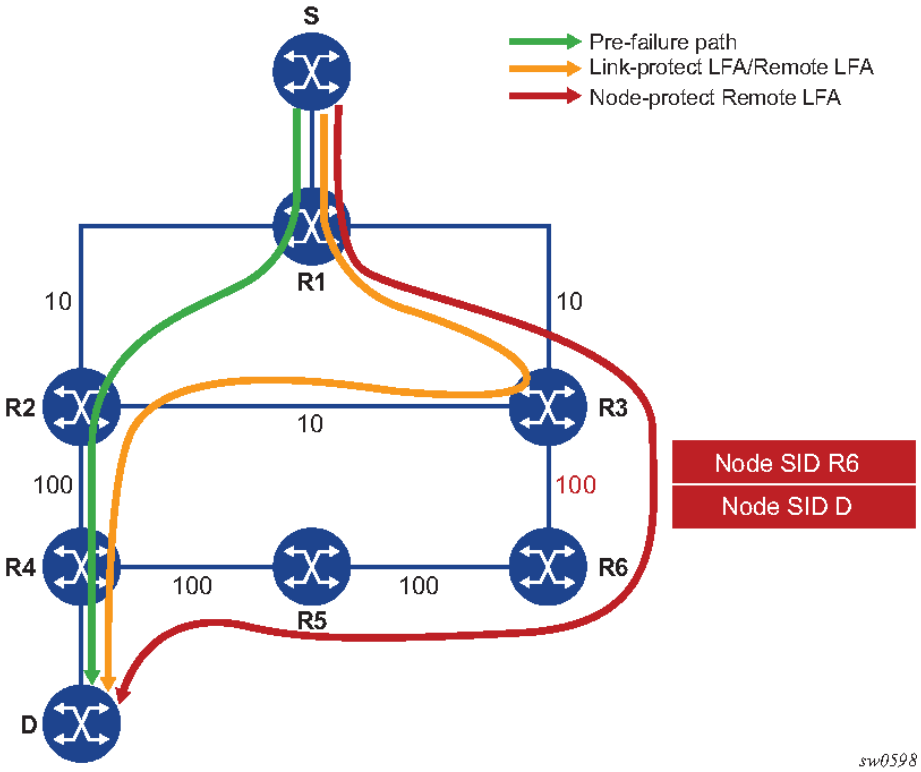
SR Linux supports the node-protect extensions to the remote LFA algorithm, as described in RFC 8102. When node protection is enabled, the router prefers a node-protect over a link-protect repair tunnel in the RLFA SPF computations. This feature protects against the failure of a downstream node. The node-protect extensions are additions to the original link-protect LFA SPF algorithm.

Remote LFA follows a similar algorithm as TI-LFA but does not limit the scope of the calculation of the extended P-Space and of the Q-Space to the post-convergence paths.

Remote LFA adds an extra forward SPF on behalf of the PQ node to ensure that, for each destination, the selected PQ-node does not use a path via the protected node.

The following figure shows an example of remote LFA for node protection.

Figure 3: Application of the remote LFA algorithm for node protection



Using the topology in the preceding figure, the node-protect remote LFA algorithm computation is performed in the following sequence.

1. **Compute extended P-Space of R1 with respect to protected node R2.**

   This is the set of nodes Yi which are reachable from R1 neighbors, other than protected node R2, without any path transiting the protected node R2.

   R1 computes an LFA SPF rooted at each of its neighbors, for example, R3, using the following equation:

   Distance_opt(R3, Yi) < Distance_opt(R3, R2) + Distance_opt(R2, Yi)

   Where *Distance_opt(A,B)* is the shortest distance between *A* and *B*.

   Nodes R3, R5, and R6 satisfy this inequality.

2. **Compute Q-space of R1 with respect to protected link R1-R2.**

   This is the set of nodes Zi from which node R2 can be reached without any path transiting the protected link R1-R2, using the following equation.

   Distance_opt(Zi, R2) < Distance_opt(Zi, R1) + Distance_opt(R1, R2)

   The reverse SPF for the Q-space calculation is the same as in the remote LFA link-protect algorithm and uses the protected node R2 as the proxy for all destination prefixes.

   This step yields nodes R3, R4, R5, and R6.

3. **For each PQ node found, run a forward SPF to each destination D.**

   This step is required to select only the subset of PQ-nodes that do not traverse protected node R2.

   Distance_opt(PQi, D) < Distance_opt(PQi, R2) + Distance_opt(R2, D)

   Of the candidates PQ nodes R3, R5, and R6, only PQ-nodes R5 and R6 satisfy this inequality. PQ-node R6 is closer in terms of IGP cost to computing router R1 and is therefore selected.

   This step of the algorithm is applied to the subset of candidate PQ-nodes out of steps 1 and 2 and to which the **max-pq-cost** parameter was already applied. This subset is further reduced in this step by retaining the candidate PQ-nodes that provide the highest coverage among all protected nodes in the topology, and the number of which does not exceed the value of the **max-pq-nodes** parameter.

4. **Select a PQ-Node.**

   If multiple PQ nodes satisfy the criteria in all the above steps, then R1 further selects the PQ node as follows.

   a. R1 selects the lowest IGP cost from R1.

   b. If more than one PQ-nodes remains, R1 selects the PQ-node reachable via the neighbor with the lowest system ID.

   c. If more than one PQ-node remains, R1 selects the PQ node with the lowest system ID.

5. **Program the remote LFA backup path.**
   For each destination prefix D, R1 programs the remote LFA backup path as follows.

   - For prefixes of R4 or downstream of R4, R1 programs a node-protect remote LFA repair tunnel to the PQ-node R6 by pushing the SID of node R6 on top of the SID for destination D and programming a next hop of R3.
   - For prefixes owned by node R2, R1 runs the link-protect remote LFA algorithm and programs a simple link-protect repair tunnel that consists of a backup next hop of R3 and pushes no additional label on top of the SID for the destination prefix.
   - Prefixes owned by nodes R3, R6, and R5 are not impacted by the failure of R2 because their primary next hop is R3.

6.4 Configuring Remote LFA

Procedure

To enable remote LFA in IS-IS, use the **remote-lfa** option.

In topologies where many eligible P or Q nodes can exist, you can specify the **max-pq-cost** to limit the P and Q node searches and reduce the number of SPF calculations. This option specifies the maximum IGP cost allowed from the point of local repair (PLR) to a candidate node for the PLR to consider the candidate as an eligible PQ node.

Example: Configure remote LFA

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance sr-isis-test
 loopfree-alternate
    network-instance default {
        protocols {
```

```
            isis {
                instance sr-isis-test {
                    loopfree-alternate {
                        remote-lfa {
                            admin-state enable
                            max-pq-cost 100
                        }
                    }
                }
            }
        }
    }
```

6.4.1 Excluding interfaces from LFA

Procedure

To exclude a network IP interface from being used as the outgoing interface for a remote LFA tunnel next hop, use the **loopfree-alternate-exclude true** command.

Note: Enabling TI-LFA in an IS-IS instance overrides the configuration of the interface **loopfree-alternate-exclude** command; that is, the TI-LFA SPF uses that interface as a backup next hop in that IS-IS instance if it matches the post-convergence next hop.

Example: Exclude interfaces from LFA

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance sr-isis-test
 interface ethernet-1/1.1
    network-instance default {
        protocols {
            isis {
                instance sr-isis-test {
                    interface ethernet-1/1.1 {
                        loopfree-alternate-exclude true
                    }
                }
            }
        }
    }
```

6.4.2 Excluding an IS-IS instance from LFA

Procedure

To exclude an entire IS-IS instance from the LFA calculations, use the **loopfree-alternate-exclude true** command. When the **level-capability** command is set to **L1L2** for the IS-IS instance, the **loopfree-alternate-exclude true** command can be applied independently to level 1 or level 2.

Example: Exclude an IS-IS instance from LFA

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance 1
    network-instance default {
        protocols {
            isis {
```

```
                instance 1 {
                    level-capability L1L2
                    level 1 {
                        loopfree-alternate-exclude true
                    }
                    level 2 {
                        loopfree-alternate-exclude false
                    }
                }
            }
        }
    }
```

6.4.3 Configuring node protection in RLFA

About this task

When node protection is enabled, the router prefers a node-protect over a link-protect repair tunnel for a prefix if both are found in the RLFA SPF computations. However, the SPF computations may still only find a link-protect repair tunnel for prefixes owned by the protected node.

Procedure

To configure the RLFA node protection feature, use the **node-protect admin-state enable** command.

Example: Configure RLFA node protection

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance sr-isis-test
 loopfree-alternate remote-lfa
    network-instance default {
        protocols {
            isis {
                instance sr-isis-test {
                    loopfree-alternate {
                        remote-lfa {
                            admin-state enable
                            max-pq-cost 100
                            node-protect {
                                admin-state enable
                                max-pq-nodes 8
                            }
                        }
                    }
                }
            }
        }
    }
```

6.5 Topology-independent LFA

Topology-independent LFA (TI-LFA) improves the protection coverage of a network topology by computing and automatically instantiating a repair tunnel to a Q-node that is not in the shortest path from the computing node. The repair tunnel uses the shortest path to the P-node and a source-routed path from the P-node to the Q-node.

In addition, the TI-LFA algorithm selects the backup path that matches the post-convergence path. This improves capacity planning in the network because traffic always flows on the same path when transitioning to the FRR next hop and then on to the new primary next hop.

At a high level, the TI-LFA link protection algorithm searches for the closest Q-node to the computing node and then selects the closest P-node to this Q-node, up to a maximum number of labels. This is performed on each of the post-convergence paths to each destination node or prefix D.

When the TI-LFA feature is enabled in IS-IS, it provides a TI-LFA link-protect backup path in IS-IS MT=0 for an SR IS-IS IPv4 or SR IS-IS IPv6 tunnel (node SID and adjacency SID) and for an IPv4 TE-Policy SR-MPLS uncolored tunnel resolving to SR-ISIS.

Supported platforms

TI-LFA is supported on the following platforms:

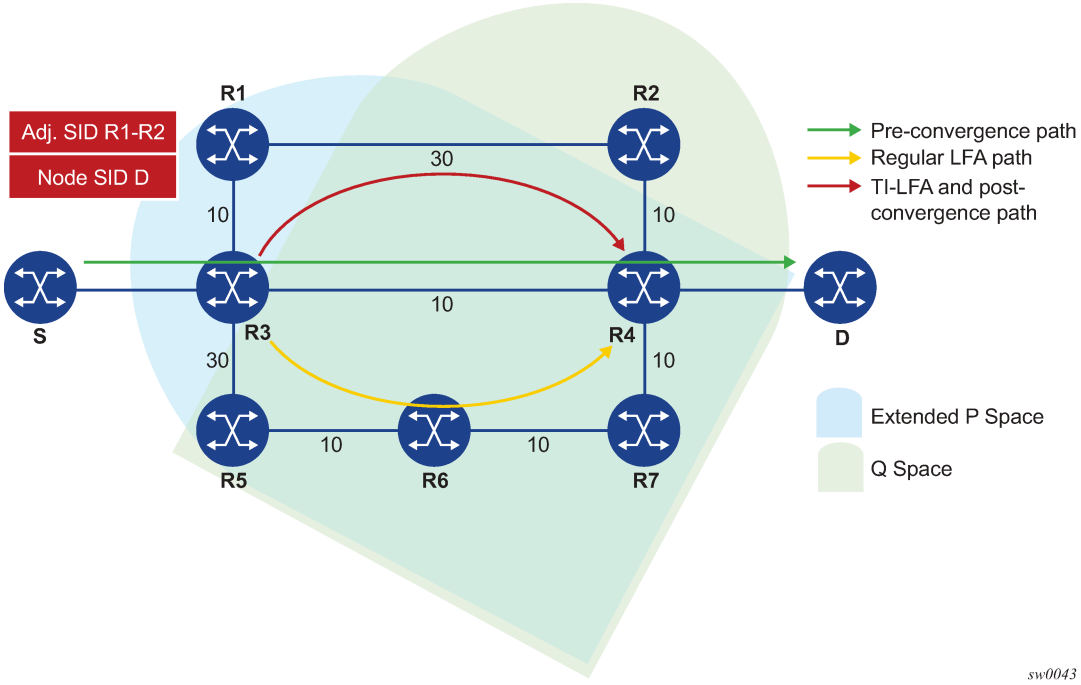
- 7250 IXR
- 7730 SXR

6.5.1 TI-LFA algorithm

For each destination node or prefix D, the TI-LFA link protection algorithm searches on each of the post-convergence paths for the closest Q-node to the computing node and then selects the closest P-node to this Q-node, up to a maximum configurable number of 3 labels.

The following figure shows a topology where router R3 computes a TI-LFA next hop for protecting link R3-R4.

Figure 4: Selecting link-protect TI-LFA backup path



In this topology, router R3 computes the link-protected TI-LFA backup path in the following order.

1. **Compute the post-convergence SPF on the topology without the protected link.**

   R3 finds a single post-convergence path to destination D via R1.

2. **Compute the extended P-Space of R3 with respect to protected link R3-R4 on the post-convergence paths.**

   This is the set of nodes Yi in the post-convergence paths that are reachable from R3 neighbors without any path transiting the protected link R3-R4.

   R3 computes an LFA SPF rooted at each of its neighbors within the post-convergence paths, that is, R1, using the following equation:

   Distance_opt(R1, Yi) < Distance_opt(R1, R3) + Distance_opt(R3, Yi)

   Where *Distance_opt(A,B)* is the shortest distance between *A* and *B*. The extended P-space calculation yields only node R1.

3. **Compute the Q-space of R3 with respect to protected link R3-R4 in the post-convergence paths.**

   This is the set of nodes Zi in the post-convergence paths from which the neighbor node R4 of the protected link, acting as a proxy for all destinations D, can be reached without any path transiting the protected link R3-R4.

   Distance_opt(Zi, R4) < Distance_opt(Zi, R3) + Distance_opt(R3, R4)

   The Q-space calculation yields nodes R2 and R4.

   This is the same computation of the Q-space performed by the remote LFA algorithm, except that the TI-LFA Q-space computation is performed only on the post-convergence paths.

4. **For each post-convergence path, search for the closest Q-node and select the closest P-node to this Q-node, up to the configurable maximum number of labels.**

   The preceding topology diagram shows a single post-convergence path and a single P-node (R1). It also shows that R2 is the closest of the two found Q-nodes to the P-Node.

   R3 installs the repair tunnel to the P-Q set and includes the node SID of R1 and the adjacency SID of the adjacency over link R1-R2 in the label stack. Because the P-node R1 is a neighbor of the computing node R3, the node SID of R1 is not needed and the label stack of the repair tunnel is compressed to the adjacency SID over link R1-R2 as shown.

   When a P-Q set is found on multiple ECMP post-convergence paths, the following selection rules are applied, in ascending order, to select a set from a single path:

   a. the lowest number of labels

   b. the next hop to the neighbor router with the lowest System ID

   c. the next hop corresponding to the Q node with the lowest System ID

   If multiple links with adjacency SIDs exist between the selected P-node and the selected Q-node, the following rules are used for link selection:

   a. the adjacency SID with the lowest metric

   b. the adjacency SID with the lowest SID value if the lowest metric is the same

6.5.2 TI-LFA feature interaction and limitations

Because the post-convergence SPF does not use paths transiting on a node in IS-IS overload, the TI-LFA backup path automatically does not transit on such a node.

6.5.3 LFA protection option applicability

Depending on the LFA options that are configured, the LFA SPF in the IGP instance runs the following algorithms in order.

1. **Compute a regular LFA for each node and prefix.**
   In this step, a computed backup next hop satisfies any applied LFA policy. This backup next hop protects that specific prefix or node in the context of SR FRR, and TE-Policy SR-MPLS FRR.
2. **If TI-LFA is enabled, run the TI-LFA algorithm for all prefixes and nodes.**
   If the **loopfree-alternate** option is enabled, a prefix or node for which a TI-LFA backup next hop is found overrides the result from step 1 in the context of SR FRR and in TE-Policy SR-MPLS FRR.

   With SR FRR, the TI-LFA next hop protects the node-SID of that prefix and any adjacency SID terminating on the same node-SID.
3. **If RLFA is enabled, run remote LFA only for the next hop of prefixes and nodes that remain unprotected after steps 1 and 2.**
   A prefix or node for which a remote LFA backup next hop is found uses it in the context of SR FRR and in TE-Policy SR-MPLS FRR.

When protecting an adjacency SID, a parallel ECMP adjacency takes precedence over any other type of LFA backup path. Applying the algorithm applicability rules results in the following selection:

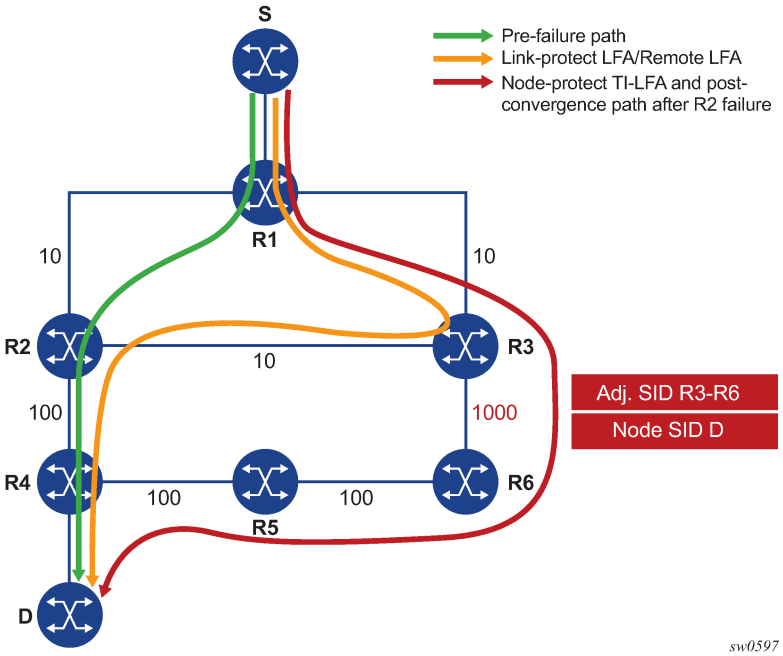
- adjacency SID of an alternate ECMP next hop
- TI-LFA backup next hop
- remote LFA backup next hop

6.5.4 TI-LFA node-protect operation

SR Linux supports the node-protect extensions to the TI-LFA algorithm as described in *draft-ietf-rtgwg-segment-routing-ti-lfa-01*. When node protection is enabled, the router prefers a node-protect over a link-protect repair tunnel in the TI-LFA SPF computations. This feature protects against the failure of a downstream node. The node-protect extensions are additions to the original link-protect LFA SPF algorithm.

The following figure shows a simple topology to illustrate the node-protect operation for TI-LFA. It uses the same topology as shown in Remote LFA node-protect operation, but the metric for link R3-R6 is modified to 1000.

Figure 5: Application of the TI-LFA algorithm for node protection



The main change as a result of the node-protect extension is that the algorithm protects a node instead of a link.

Using the topology in the preceding figure, the node protection computation is performed in the following sequence.

1. **Compute the post-convergence SPF on the topology without the protected node.**

   In the preceding figure, R1 computes TI-LFA on the topology without the protected node R2 and finds a single post-convergence path to destination D via R3 and R6.

   Prefixes owned by all other nodes in the topology have a post-convergence path via R3 and R6 except for prefixes owned by node R2. The latter uses the link R3-R2 and they can only benefit from link protection.

2. **Compute the extended P-Space of R1 with respect to protected node R2 on the post-convergence paths.**

   This is the set of nodes Yi in the post-convergence paths that are reachable from R1 neighbors, other than protected node R2, without any path transiting the protected node R2.

   R1 computes an LFA SPF rooted at each of its neighbors within the post-convergence paths. For example, R1 uses using the following calculation to compute the LFA SPF for R3:

   Distance_opt(R3, Yi) < Distance_opt(R3, R2) + Distance_opt(R2, Yi)

   Where *Distance_opt(A,B)* is the shortest distance between *A* and *B*.

   The extended P-space calculation yields node R3 only.

3. **Compute the Q-space of R1 with respect to protected link R1-R2 on the post-convergence paths.**

   This is the set of nodes Zi in the post-convergence paths from which node R2 can be reached without any path transiting the protected link R1-R2, using the following equation:

Distance_opt(Zi, R2) < Distance_opt(Zi, R1) + Distance_opt(R1, R2)

The reverse SPF for the Q-space calculation is the same as in the link-protect algorithm and uses the protected node R2 as the proxy for all destination prefixes. To compute the Q space with respect to the protected node R2 instead of link R1-R2, a reverse SPF would have to be performed for each destination D which is very costly and not scalable. However, this means the path from the Q-node to the destination D or the path from the P-node to the Q-node is not guaranteed to avoid the protected node R2. The intersection of the Q-space with post-convergence path is modified in the next step to mitigate this risk.

This step yields nodes R3, R4, R5, and R6.

4. **For each post-convergence path, search for the closest Q-node to destination D and select the closest P-node to this Q-node, up to the configurable maximum number of labels.**

   This step yields the following P-Q sets, depending on the value of the **ti-lfa max-sr-policy-lfa-labels** parameter:

   - **max-sr-policy-lfa-labels=0**

     R3 is the closest Q-node to the destination D and R3 is the only P-node. This case results in link protection via PQ-node R3.

   - **max-sr-policy-lfa-labels=1**

     R6 is the closest Q-node to the destination D and R3 is the only P-node. The repair tunnel for this case uses the SID of the adjacency over link R3-R6 and is shown in the topology diagram.

   - **max-sr-policy-lfa-labels=2 (default)**

     R5 is the closest Q-node to the destination D and R3 is the only P-node. The repair tunnel for this case uses the SIDs of the adjacencies over links R3-R6 and R6-R5.

   - **max-sr-policy-lfa-labels=3**

     R4 is the closest Q-node to the destination D and R3 is the only P-node. The repair tunnel for this case uses the SIDs of the adjacencies over links R3-R6, R6-R5, and R5-R4.

   This step of the algorithm is modified from link protection, which prefers Q-nodes that are the closest to the computing router R1. This is to minimize the probability that the path from the Q-node to the destination D, or the path from the P-node to the Q-node, goes via the protected node R2 as described in step 2. However, there is still a probability that the found P-Q set achieves link protection only.

5. **Select the P-Q Set.**

   If a candidate P-Q set is found on each of the multiple ECMP post-convergence paths in step 4, the following selection rules are applied in ascending order to select a single set:

   a. the lowest number of labels

   b. the lowest next-hop router ID

   c. the lowest subinterface index if the same as the next-hop router ID

   If multiple parallel links with adjacency SID exist between the P- and Q-nodes of the selected P-Q set, the following rules are used to select one of them:

   a. the adjacency SID with lowest metric

   b. the adjacency SID with the lowest SID value, if the same as the lowest metric

6. **Program the TI-LFA repair tunnel.**
   For each destination prefix D, R1 programs the TI-LFA repair tunnel (**max-sr-policy-lfa-labels**=*1*):

- For prefixes other than those owned by node R2, R1 programs a node-protect repair tunnel to the P-Q pair R3-R6 by pushing the SID of adjacency R3-R6 on top of the SID for destination D and programming a next hop of R3.
- For prefixes owned by node R2, R1 runs the link-protect TI-LFA algorithm and programs a simple link-protect repair tunnel, which consists of a backup next hop of R3 and pushes no additional label on top of the SID for the destination prefix.

6.5.4.1 TI-LFA and remote LFA node protection feature interaction and limitations

The order of activation of the various LFA types on a per prefix basis is as follows: TI-LFA, followed by base LFA, followed by remote LFA. See LFA protection option applicability for more information about the order of activation.

Node protection is enabled for TI-LFA and remote LFA separately. The base LFA prefers node protection over link protection.

The order of activation of the LFA types supersedes the protection type (node versus link). Consequently, a prefix can be programmed with a link-protect backup next hop by the more preferred LFA type. For example, a prefix is programmed with the only link-protect backup next hop found by the base LFA when a node-protect remote LFA next hop exists.

6.6 Configuring TI-LFA

About this task

The **ti-lfa** option in IS-IS provides a TI-LFA link-protect backup path in IS-IS MT=0 for an SR IS-IS IPv4 and IPv6 tunnel (node SID and adjacency SID) and for an IPv4 TE-Policy SR-MPLS tunnel .

Procedure

To enable TI-LFA in an IS-IS instance, use the **loopfree-alternate ti-lfa** command.

Example: Enable TI-LFA on an ISIS instance

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance sr-isis-test
 loopfree-alternate ti-lfa
    network-instance default {
        protocols {
            isis {
                instance sr-isis-test {
                    loopfree-alternate {
                        ti-lfa {
                            admin-state enable
                            max-sr-policy-lfa-labels 1
                        }
                    }
                }
            }
        }
    }
```

7 BGP shortcuts configuration over segment routing tunnels

When you have segment routing enabled in your network, you can specify a segment routing tunnel as the next hop for a BGP route. This capability is referred to as BGP shortcuts over segment routing.

Uncolored SR-MPLS TE-Policy tunnels (`sr-mpls-uncolored`) in TTM can resolve BGP IPv4 routes.

IPv6 NHLFEs can be used for both BGP IPv4 and BGP IPv6 routes.

To configure BGP shortcuts, you must configure the BGP protocol with the allowed tunnel types and the required tunnel resolution mode.

7.1 Configuring BGP shortcuts over segment routing

Procedure

Step 1. In the default network instance, define the tunnel-resolution mode for the BGP protocol.

This setting determines the order of preference and the fallback when using tunnels in the tunnel table instead of routes in the FIB. Available options are as follows:

- **require**
  requires tunnel table lookup only
- **prefer**
  prefers tunnel table lookup over FIB lookup
- **disabled (default)**
  performs FIB lookup only

Step 2. Set the allowed tunnel types for next-hop resolution.

Example: Configure IPv4 BGP shortcuts

The following example shows the BGP next-hop resolution configuration to allow IPv4 SR-ISIS tunnels, with the tunnel mode set to `prefer`.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast ipv4-
unicast next-hop-resolution ipv4-next-hops tunnel-resolution
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    ipv4-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                tunnel-resolution {
                                    mode prefer
                                    allowed-tunnel-types [
                                        sr-isis
                                    ]
                                }
                            }
```

```
                    }
                }
            }
        }
    }
```

Example: Configure IPv6 BGP shortcuts

The following example shows the BGP next-hop resolution configuration to allow IPv6 SR-ISIS tunnels, with the tunnel mode set to `prefer`.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols bgp afi-safi ipv6-unicast ipv6-
unicast next-hop-resolution ipv6-next-hops tunnel-resolution
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv6-unicast {
                    ipv6-unicast {
                        next-hop-resolution {
                            ipv6-next-hops {
                                tunnel-resolution {
                                    mode prefer
                                    allowed-tunnel-types [
                                        sr-isis
                                    ]
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

8 Segment routing display commands

SR Linux supports display commands to provide operational information about segment routing, including the following:

- SID database
- label block information
- tunnel-table entries

8.1 Displaying the SID database

Procedure

Use the **info from state** command to display information about the SID database.

Example: Display the global SID database

```
--{ candidate shared default }--[ ]--
# info with-context from state network-instance default segment-routing mpls sid-database
    network-instance default {
        segment-routing {
            mpls {
                sid-database {
                    prefix-sid 10.20.1.1/32 sid-label-value 3010 protocol isis protocol-instance 0
 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 10.20.1.2/32 sid-label-value 3020 protocol isis protocol-instance 0
 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 10.20.1.3/32 sid-label-value 3030 protocol isis protocol-instance 0
 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 10.20.1.4/32 sid-label-value 3040 protocol isis protocol-instance 0
 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 10.20.1.5/32 sid-label-value 3050 protocol isis protocol-instance 0
 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 10.20.1.6/32 sid-label-value 3060 protocol isis protocol-instance 0
 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 2001:db8:20:1::1/128 sid-label-value 3610 protocol isis protocol-
instance 0 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 2001:db8:20:1::2/128 sid-label-value 3620 protocol isis protocol-
instance 0 protocol-multi-topology 0 algorithm 0 {
                        active true
```

```
                    }
                    prefix-sid 2001:db8:20:1::3/128 sid-label-value 3630 protocol isis protocol-
instance 0 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 2001:db8:20:1::4/128 sid-label-value 3640 protocol isis protocol-
instance 0 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 2001:db8:20:1::5/128 sid-label-value 3650 protocol isis protocol-
instance 0 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                    prefix-sid 2001:db8:20:1::6/128 sid-label-value 3660 protocol isis protocol-
instance 0 protocol-multi-topology 0 algorithm 0 {
                        active true
                    }
                }
            }
        }
    }
```

Example: Display the IS-IS prefix SID database

```
--{ candidate shared default }--[ ]--
# info with-context from state network-instance default protocols isis instance default segment-
routing mpls sid-database
    network-instance default {
        protocols {
            isis {
                instance default {
                    segment-routing {
                        mpls {
                            sid-database {
                                prefix-sid 10.20.1.1/32 sid-label-value 41000 multi-topology-id 0
 algorithm 0 {
                                    active false
                                    prefix-conflict false
                                    sid-conflict false
                                    sid-out-of-range false
                                    source-router 0100.2000.1001 level-number 1 {
                                        local-system false
                                        flags {
                                            re-advertised false
                                            node-sid true
                                            penultimate-hop-popping true
                                            explicit-null false
                                            local false
                                        }
                                    }
                                    source-router 0100.2000.1001 level-number 2 {
                                        local-system false
                                        flags {
                                            re-advertised false
                                            node-sid true
                                            penultimate-hop-popping true
                                            explicit-null false
                                            local false
                                        }
                                    }
                                    source-router 0100.2000.1002 level-number 2 {
                                        local-system false
                                        flags {
```

```
                                        re-advertised true
                                        node-sid true
                                        penultimate-hop-popping true
                                        explicit-null false
                                        local false
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

8.2 Displaying label block information

Procedure

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

Example:

```
--{ * candidate shared default }--[  ]--
# info with-context from state system mpls label-ranges
    system {
        mpls {
            label-ranges {
                static {
                    name srgb-range-1
                    start-label 16001
                    end-label 16999
                    allocated-labels 10
                    free-labels 989
                    status ready
                    }
                }
            }
        }
    }
```

8.3 Displaying tunnel table entries

Procedure

Use the **show network-instance default tunnel-table** command to display information about segment routing tunnel table entries. You can adjust the output of the report to filter by address type, encapsulation type, tunnel type, and destination prefix.

Example: Show all tunnel-table entries

```
--{ running }--[  ]--
# show network-instance default tunnel-table all
-----------------------------------------------------------------------------------------------------
```

```
IPv4 tunnel table of network-instance "default"
------------------------------------------------------------------------------------------------------------
+------------+------+-------+------+---+------+----------+-------------+-----------+-----------------+
|IPv4 Prefix |Encaps|Tunnel |Tunnel|FIB|Metric|Preference| Last Update | Next-hop  |    Next-hop     |
|            |Type  |Type   |ID    |   |      |          |             | (Type)    |                 |
+============+======+=======+======+===+======+==========+=============+===========+=================+
|10.20.1.1/32| mpls |sr-isis|13010 | Y |  10  | 11       | 2021-11-10T | 10.10.1.1 | ethernet-1/33.1 |
|            |      |       |      |   |      |          |14:17:32.202Z| (mpls)    |                 |
|10.20.1.3/32| mpls |sr-isis|13030 | Y |  10  | 11       | 2021-11-10T | 10.10.3.3 | ethernet-1/34.1 |
|            |      |       |      |   |      |          |14:18:02.299Z| (mpls)    |                 |
|10.20.1.4/32| mpls |sr-isis|13040 | Y |  10  | 11       | 2021-11-10T | 10.10.4.4 | ethernet-1/8.1  |
|            |      |       |      |   |      |          |14:18:26.729Z| (mpls)    |                 |
|10.20.1.5/32| mpls |sr-isis|13050 | Y |  10  | 11       | 2021-11-10T | 10.10.23.5| ethernet-1/9.1  |
|            |      |       |      |   |      |          |14:19:02.343Z| (mpls)    |                 |
|10.20.1.6/32| mpls |sr-isis|13060 | Y |  10  | 11       | 2021-11-10T | 10.10.14.6| ethernet-1/11.1 |
|            |      |       |      |   |      |          |14:19:38.140Z| (mpls)    |                 |
+------------+------+-------+------+---+------+----------+-------------+-----------+-----------------+
------------------------------------------------------------------------------------------------------------
5 SR-ISIS tunnels, 5 active, 0 inactive
------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------
IPv6 tunnel table of network-instance "default"
------------------------------------------------------------------------------------------------------------
+------------+------+-------+------+---+------+-----+-------------+------------------+---------------+
|IPv6 Prefix |Encaps|Tunnel |Tunnel|FIB|Metric|Prefe| Last Update |     Next-hop     |   Next-hop    |
|            |Type  |Type   |ID    |   |      |rence|             |     (Type)       |               |
+============+======+=======+======+===+======+=====+=============+==================+===============+
|2001:db8:20:| mpls |sr-isis|13610 | Y | 10   | 11  |2021-11-10T  |fe80::201:        |ethernet-1/33.1|
|1::1/128    |      |       |      |   |      |     |14:17:32.204Z|1ff:feff:20(mpls) |               |
|2001:db8:20:| mpls |sr-isis|13630 | Y | 10   | 11  |2021-11-10T  |fe80::201:        |ethernet-1/34.1|
|1::3/128    |      |       |      |   |      |     |14:18:03.316Z|3ff:feff:21(mpls) |               |
|2001:db8:20:| mpls |sr-isis|13640 | Y | 10   | 11  |2021-11-10T  |fe80::201:        |ethernet-1/8.1 |
|1::4/128    |      |       |      |   |      |     |14:18:31.638Z|4ff:feff:20(mpls) |               |
|2001:db8:20:| mpls |sr-isis|13650 | Y | 10   | 11  |2021-11-10T  |fe80::201:        |ethernet-1/9.1 |
|1::5/128    |      |       |      |   |      |     |14:19:04.882Z|5ff:feff:20(mpls) |               |
|2001:db8:20:| mpls |sr-isis|13660 | Y | 10   | 11  |2021-11-10T  |fe80::201:        |ethernet-1/11.1|
|1::6/128    |      |       |      |   |      |     |14:19:43.545Z|6ff:feff:4(mpls)  |               |
+------------+------+-------+------+---+------+-----+-------------+------------------+---------------+
------------------------------------------------------------------------------------------------------------
5 SR-ISIS tunnels, 5 active, 0 inactive
------------------------------------------------------------------------------------------------------------
--{  running }--[  ]--
```

Example: Show IPv4 tunnel-table entries

```
--{ running }--[  ]--
# show network-instance default tunnel-table ipv4
------------------------------------------------------------------------------------------------------------
IPv4 tunnel table of network-instance "default"
------------------------------------------------------------------------------------------------------------
+------------+------+-------+------+---+------+----------+-------------+-----------+-----------------+
|IPv4 Prefix |Encaps|Tunnel |Tunnel|FIB|Metric|Preference| Last Update | Next-hop  |    Next-hop     |
|            |Type  |Type   |ID    |   |      |          |             | (Type)    |                 |
+============+======+=======+======+===+======+==========+=============+===========+=================+
|10.20.1.1/32| mpls |sr-isis|13010 | Y | 10   | 11       | 2021-11-10T | 10.10.1.1 | ethernet-1/33.1 |
|            |      |       |      |   |      |          |14:17:32.202Z| (mpls)    |                 |
|10.20.1.3/32| mpls |sr-isis|13030 | Y | 10   | 11       | 2021-11-10T | 10.10.3.3 | ethernet-1/34.1 |
|            |      |       |      |   |      |          |14:18:02.299Z| (mpls)    |                 |
|10.20.1.4/32| mpls |sr-isis|13040 | Y | 10   | 11       | 2021-11-10T | 10.10.4.4 | ethernet-1/8.1  |
|            |      |       |      |   |      |          |14:18:26.729Z| (mpls)    |                 |
|10.20.1.5/32| mpls |sr-isis|13050 | Y | 10   | 11       | 2021-11-10T | 10.10.23.5| ethernet-1/9.1  |
|            |      |       |      |   |      |          |14:19:02.343Z| (mpls)    |                 |
|10.20.1.6/32| mpls |sr-isis|13060 | Y | 10   | 11       | 2021-11-10T | 10.10.14.6| ethernet-1/11.1 |
```

```
|            |      |       |      |   |      |          |14:19:38.140Z| (mpls)    |                |
+------------+------+-------+------+---+------+----------+-------------+-----------+----------------+
-------------------------------------------------------------------------------------------------------
5 SR-ISIS tunnels, 5 active, 0 inactive
-------------------------------------------------------------------------------------------------------
```

Example: Show IPv6 tunnel-table entries

```
--{ running }--[  ]--
# show network-instance default tunnel-table ipv6
-------------------------------------------------------------------------------------------------------
IPv6 tunnel table of network-instance "default"
-------------------------------------------------------------------------------------------------------
+------------+------+-------+------+---+------+-----+-------------+-----------------+---------------+
|IPv6 Prefix |Encaps|Tunnel |Tunnel|FIB|Metric|Prefe|Last Update  |    Next-hop     |   Next-hop    |
|            |Type  |Type   |ID    |   |      |rence|             |    (Type)       |               |
+============+======+=======+======+===+======+=====+=============+=================+===============+
|2001:db8:20:|mpls  |sr-isis|13610 | Y | 10   | 11  |2021-11-10T  |fe80::201:       |ethernet-1/33.1|
|1::1/128    |      |       |      |   |      |     |14:17:32.202Z|1ff:feff:20(mpls)|               |
|2001:db8:20:|mpls  |sr-isis|13630 | Y | 10   | 11  |2021-11-10T  |fe80::201:       |ethernet-1/34.1|
|1::3/128    |      |       |      |   |      |     |14:18:02.299Z|3ff:feff:21(mpls)|               |
|2001:db8:20:|mpls  |sr-isis|13640 | Y | 10   | 11  |2021-11-10T  |fe80::201:       |ethernet-1/8.1 |
|1::4/128    |      |       |      |   |      |     |             |4ff:feff:20(mpls)|               |
|2001:db8:20:|mpls  |sr-isis|13650 | Y | 10   | 11  |2021-11-10T  |fe80::201:       |ethernet-1/9.1 |
|1::5/128    |      |       |      |   |      |     |             |5ff:feff:20(mpls)|               |
|2001:db8:20:|mpls  |sr-isis|13660 | Y | 10   | 11  |2021-11-10T  |fe80::201:       |ethernet-1/11.1|
|1::6/128    |      |       |      |   |      |     |             | 6ff:feff:4(mpls)|               |
+------------+------+-------+------+---+------+-----+-------------+-----------------+---------------+
-------------------------------------------------------------------------------------------------------
5 SR-ISIS tunnels, 5 active, 0 inactive
-------------------------------------------------------------------------------------------------------
```

Example: Show IPv4 tunnel-table entries by destination prefix

```
--{ running }--[  ]--
# show network-instance default tunnel-table ipv4 10.20.1.6/32 type sr-isis detail
-------------------------------------------------------------------------------------------------------
Show report for network instance "default" tunnel table
-------------------------------------------------------------------------------------------------------
=======================================================================================================
Destination        : 10.20.1.6/32
Encapsulation      : mpls
Tunnel Type        : sr-isis
Metric             : 10
Preference         : 11
Last Update        : 2021-11-10T14:19:38.140Z
FIB Status         : active
Next-hops
  10.10.14.6 (mpls) via [ethernet-1/11.1]
      pushed MPLS labels : [53060]
=======================================================================================================
```

Example: Show IPv6 tunnel-table entries by destination prefix

```
# show network-instance default tunnel-table ipv6 2001:db8:10:20:1::6/128 detail

-------------------------------------------------------------------------------------------------------
Show report for network instance "default" tunnel table
-------------------------------------------------------------------------------------------------------
=======================================================================================================
Destination        : 2001:db8:10:20:1::6/128
```

```
Encapsulation      : mpls
Tunnel Type        : sr-isis
Metric             : 10
Preference         : 11
Last Update        : 2021-11-10T14:19:43.545Z
FIB Status         : active
Next-hops
  2001:db8:201:6ff:feff:4 (mpls) via [ethernet-1/11.1]
      pushed MPLS labels : [53660]
===============================================================================================
```

9 LSP ping and trace for segment routing tunnels

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

- **tools oam lsp-ping sr-isis prefix-sid** <*prefix*>
- **tools oam lsp-trace sr-isis prefix-sid** <*prefix*>

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

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

In the case of ECMP, even when the destination IP is configured, the SR Linux node may not exercise all NHLFEs.

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

10 Traffic Engineering

Note: Traffic Engineering is supported on 7250 IXR and 7730 SXR platforms.

Traffic engineering (TE) is a method for efficiently routing network traffic to maximize throughput and minimize delay.

Without TE, network traffic follows the shortest or most efficient paths calculated by IGP (IS-IS/OSPF) or BGP protocols. These paths, however, disregard link state attributes, including bandwidth utilization, individual link delays, and other real-time network conditions, which can result in congestion.

IGP protocols are extended with new Type Length Values (TLVs) that specify these link and node state attributes. IGP TE extensions are responsible for advertising the TE attributes within the IGP domain. Currently, SR Linux supports only IS-IS extensions for Traffic Engineering (TE).

Uncolored SR-MPLS TE policies define the rules for traffic engineering in an SR-enabled network.

SR Linux supports the following features to enable TE within the SR framework:

- Support for sending and receiving TE-related TLVs and sub-TLVs
- Configurable IPv4/v6 TE identifiers
- Configurable TE interface
- Configurable TE metrics for each TE interface
- Configurable administrative group (**admin-group**) for each TE interface
- Configurable Shared Risk Link Group (SRLG) membership for each TE interface
- Configurable static delay for each TE interface

10.1 Configuring TE identifier

Procedure

SR Linux supports configuring routable IPv4 and IPv6 router addresses to identify the router uniquely in a TE domain.

Example: Configuring an IPv4 TE identifier

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering ipv4-te-router-id
    network-instance default {
        traffic-engineering {
            ipv4-te-router-id 10.1.1.1
        }
    }
```

Example: Configuring an IPv6 TE identifier

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering ipv6-te-router-id
```

```
    network-instance default {
        traffic-engineering {
            ipv6-te-router-id 2001:db8::1
        }
    }
```

10.2 Configuring TE interface

Procedure

The configurable TE interface allows the selection of which interfaces are used for traffic engineering. To configure TE on an interface, use the command `network-instance traffic-engineering interface`. When TE is enabled, the TE-related Type-Length-Value (TLV) and sub-TLV fields are incorporated into the OSPF Link State Advertisements (LSAs) or IS-IS Link State Packets (LSPs) generated by these IGPs.

Example: Enabling TE on an interface

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering
    network-instance default {
        traffic-engineering {
            interface srl_test_interface {
            }
        }
    }
```

Example: Configuring TE using interface-ref

The `interface-ref` parameter in TE configuration commands allows you to reference an `interface` or `subinterface` instead of configuring TE for an interface. Based on the specified `interface` and `subinterface` leaves, the interface is uniquely referenced.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering interface srl_test_
interface interface-ref
    network-instance default {
        traffic-engineering {
            interface srl_test_interface {
                interface-ref {
                    interface ethernet-1/1
                    subinterface 0
                }
            }
        }
    }
```

10.3 Configuring TE metric

Procedure

Configuring TE metrics for each TE interface influences routing decisions based on network conditions and policies. When the TE metric is selected for an LSP, the shortest path calculation selects the path based

on the TE metric instead of the IGP metric after applying the TE constraints. Both the TE and IGP metrics are advertised by IGP protocols for each link in the network. The TE metric is part of the traffic engineering extensions of IGP protocols. SR Linux supports only IS-IS TE extensions and implements the TE metric as defined in RFC 8570. The `te-metric` value is advertised in sub-TLV (type 18) of the Extended IS Reachability TLV (type 22).

Example

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering interface srl_test_
interface
    network-instance default {
        traffic-engineering {
            interface srl_test_interface {
                te-metric 564
            }
        }
    }
```

10.4 Configuring TE admin-groups

Procedure

SR Linux supports Administrative Groups (AGs) as defined in RFC 5305. Administrative groups are manually assigned attributes that describe the color of the links. The administrative group, also referred to as link colors or resource classes, helps to identify and manage links that belong to the same administrative or resource class and is used to implement policy/constraint-based LSPs. The `admin-group` is advertised in sub-TLV (type 3) of the Extended IS Reachability TLV (type 22).

Example: Configuring admin-groups

Configuring an admin group for TE interface allows network administrators to classify and control routing decisions based on specific criteria or policies.
In this example, an `admin-group` named `blue` is configured.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering admin-groups
    network-instance default {
        traffic-engineering {
            admin-groups {
                group blue {
                }
            }
        }
    }
```

Example: Configuring admin-group bit position

The administrative group contains a 4-octet bit mask assigned by the administrator. The `bit-position` value corresponds to the location of the bit set starting from the least significant bit in the `admin-group` bit mask. Each set bit corresponds to the administrative group assigned to an interface. An `admin-group` can be associated with only one `bit-position`.

In this example, the `bit-position two` corresponds to the blue `admin-group`.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering admin-groups group blue
 bit-position
    network-instance default {
        traffic-engineering {
            admin-groups {
                group blue {
                    bit-position 2
                }
            }
        }
    }
```

Example: Associating admin-groups with a TE interface

By associating admin groups to each TE interface, you can identify the groups to which an interface belongs. The IGPs use the group information to construct link-state packets that are flooded throughout the network, providing information to all network nodes. An interface can be associated with multiple admin groups. For information about configuring the TE interface, see Configuring TE interface.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering interface srl_test_
interface
    network-instance default {
        traffic-engineering {
            interface srl_test_interface {
                te-metric 564
                admin-group [
                    blue
                ]
            }
        }
    }
```

10.5 Configuring Shared Risk Link Groups and SRLG membership

Procedure

Configuring Shared Risk Link Groups (SRLGs) to group links with common risk factors ensures alternate path selection for reliability. SRLGs allow users to establish a backup secondary label switched path (LSP) or a fast-reroute (FRR) LSP, which is disjoint from the primary LSP. A backup path is an alternative route for network traffic when the primary path becomes unavailable due to network issues. Links that are members of the same SRLG represent resources that share the same risk. In an SRLG, if one link fails, all the other links in that SRLG also fail. For example, sub-interfaces go down if their main interface goes down because of shared risks. SR Linux supports SRLGs as defined in RFC 4205. SRLGs (`shared-risk-link-groups`) are used to identify the links that belong to the same SRLG. The `shared-risk-link-groups` is advertised in new Shared Risk Link Group Type Length Values (TLVs) (type 138).

Example: Configuring shared-risk-link-groups

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering shared-risk-link-groups
```

```
    network-instance default {
        traffic-engineering {
            shared-risk-link-groups {
                group gray {
                }
            }
        }
    }
```

Example: Configuring shared-risk-link-groups value

Each SRLG is represented by a specific group value (`value`). The group value associated with `shared-risk-link-groups` uniquely identifies an SRLG. In this example, SRLG group `gray` is assigned a group `value` of one.

```
--{ * candidate shared default }--[  ]--#
info with-context network-instance default traffic-engineering shared-risk-link-groups
 group gray
    network-instance default {
        traffic-engineering {
            shared-risk-link-groups {
                group gray {
                    value 1
                }
            }
        }
    }
```

Example: Associating SRLG membership with a TE interface

An SRLG membership lists all SRLGs associated with the interface. Backup paths are computed based on the SRLG membership information exchanged between routers. Routers avoid using links belonging to the same SRLG when creating backup paths. In this example, the **srl_test_interface** interface is linked to SRLG membership, which includes the **black**, **gray**, and **red** SRLGs.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering interface srl_test_
interface srlg-membership
    network-instance default {
        traffic-engineering {
            interface srl_test_interface {
                srlg-membership [
                    black
                    gray
                    red
                ]
            }
        }
    }
```

10.6 Advertising link delay with IS-IS

Procedure

Note: This feature is supported on 7250 IXR Gen 2, 7250 IXR Gen 2c+, 7250 IXR Gen 3, and 7730 SXR platforms.

The link delay is an interface attribute described in RFC8570 and is advertised by SR Linux using the Minimum/Maximum Unidirectional Link Delay TLV (type 34).

The static delay represents a forward-path metric, measured in microseconds, between two routers. The static delay can be configured under `network-instance traffic-engineering interface` with a configurable range of 1 to 16777215 microseconds.

When you configure interface delay for an IGP, the IGP automatically includes the delay TLV34 in its Link State Packet (LSP) when TE extensions are enabled. For more information about configuring an interface delay for an IS-IS instance, see Configuring interface delay. SR Linux supports the configuration of both `static` and `dynamic` interface delay measurements. The IGP must determine which of these delay measurments to advertise. There are four potential options for the IGP to consider when deciding between the two delay measurements associated with a particular interface.

Table 7: Interface delay and link delay advertisement

Interface delay	Link delay advertised
`static`	Only the statically configured link delay is advertised. If no `static` delay is configured, the Type 34 TLV is not generated for the link.
`static-preferred`	The IGP advertises the statically configured link delay and defaults to the dynamically measured delay if no `static` delay is configured. If neither a `static` delay is configured nor a `dynamic` delay is reported, the Type 34 TLV is not generated for the link.
`dynamic`	Only the dynamically measured link delay is advertised. If no `dynamic` delay is reported, the Type 34 TLV is not generated for the link.
`dynamic-preferred`	The IGP advertises the dynamically measured link delay and defaults to the `static` link delay if no `dynamic` delay is configured. If neither a `static` delay is configured nor a `dynamic` delay is reported, the Type 34 TLV is not generated for the link.

SR Linux supports the advertising of link delay using either the legacy-link-attribute-advertisement or application-specific link attributes (ASLA) encodings, or both. The routers store the link delay parameters in the shared Link State Database (LSDB).

Example: Configuring static delay

The following example shows the configuration of static delay.

```
--{ * candidate shared default }--[  ]--
# info with-context interface ethernet-1/1 subinterface 0
    interface ethernet-1/1 {
        subinterface 0 {
            admin-state enable
            unidirectional-link-delay {
                static-delay 1234
            }
            ipv4 {
                admin-state enable
                address 192.168.0.0/31 {
                }
            }
            ipv6 {
                admin-state enable
                address 2002::c0a8:0/31 {
                }
            }
        }
    }
```

Example: Advertising of link delay

In the following config example, the statically configured link delay is advertised.

```
--{ + candidate shared default }--[  ]--
info with-context network-instance default protocols isis instance srl_test_instance
    network-instance default {
        protocols {
            isis {
                instance srl_test_instance {
                    interface ethernet-1/1.0 {
                        delay {
                            delay-selection static
                        }
                    }
                }
            }
        }
    }
```

Note: The TE attributes are only supported for Ethernet and LAG interfaces in network instances of type `default`, not for IP-VRF (`non-default network-instance`).

11 TE-Policy

TE policies define the rules for traffic engineering in an SR-enabled network. They determine how traffic should be routed across the network based on various constraints.

The TE policy is a set of segment lists, where each segment list is a collection of segment IDs (SID). A segment list can be constructed with multiple SIDs to meet the traffic engineering requirement for a given tunnel. SID specifies a path from source to destination, instructing routers to follow the specified path instead of the shortest route calculated by the IGP. After a data packet is imported into an SR-MPLS TE policy, the ingress (head-end) adds the SID list to the packet, and other routers on the network execute the instructions embedded in the SID list.

TE policies can be categorized into uncolored and colored types.

- Uncolored SR-MPLS TE policy (`sr-mpls-uncolored`): A single active LSP is represented as a segment list under the TE policy. Ideally, the primary segment list is used, with the standby as an alternative. As a last resort, the secondary segment list becomes active to provide a TE path to a given destination. Note that in 7x50 SR OS routers, uncolored SR-MPLS TE-Policy segment lists are referred to as SR-TE LSPs.
- SR-MPLS colored TE policy (`sr-mpls-colored`): Segment lists based on labels or simply segment lists constitute candidate paths, where each path is configured using a color and an endpoint. This is similar to SR Policies, where colors and endpoints are used to identify candidate paths. Note that in 7x50 SR OS routers, colored TE-Policy are referred to as SR policies.

11.1 Uncolored SR-MPLS TE-Policy

Uncolored SR-MPLS TE-policy refers to a type of TE-policy used in an SR-enabled network that does not use color to identify the policy. A single active LSP is represented as a segment list under the TE policy. An uncolored SR-MPLS TE LSP supports a primary path, with Fast Reroute (FRR) backup, and one or more secondary paths. A secondary path can be configured as standby.

SR Linux supports configuration of Seamless Bidirectional Forwarding Detection (S-BFD) to validate the operational liveliness of associated TE-Policy segment lists. Seamless BFD can be enabled on segment lists to detect liveliness and trigger switchover between active and standby segment lists. For more information, see the "Seamless Bidirectional Forwarding Detection (S-BFD)" chapter of the *SR Linux OAM and Diagnostics Guide*.

Supported platforms

Uncolored SR-MPLS TE-Policy is supported on the following platforms:

- 7250 IXR Gen 2
- 7250 IXR Gen 2c+
- 7730 SXR
- 7250 IXR Gen 3

11.1.1 Basic concepts

The following terms are essential for understanding the structure of an SR-MPLS TE policy and the interrelationship between policies.

Segment list

A segment list is an ordered sequence of SIDs that steers traffic along a specific path when translated to a data path encapsulation. You can configure segment lists directly, derive them from other configurations (for example, an explicit path specifying IP addresses or mpls labels), or compute them.

Binding Segment Identifier (BSID)

A BSID is an identifier that represents an SR-MPLS TE policy. When a packet with a BSID is received, the router selects an SR-MPLS TE policy based on the BSID to route the packet. With BSIDs, complex path policies can be encapsulated into single identifiers thus simplifying their application.

Endpoint

An endpoint defines the destination of the SR-MPLS TE policy. It is the IP address of the final destination node or router for the traffic. This is a mandatory parameter for SR-MPLS TE policy configuration.

Explicit path

The SR-MPLS TE policies can define explicit paths, where the exact sequence of SIDs is specified manually. The explicit path can be configured based on an IP address or MPLS label. The explicit path can include strict, loose, or a combination of both types of hops.

Dynamic path

A dynamic path provides an optimization objective and a set of constraints. The ingress Label Edge Router (iLER) can compute dynamic paths using local CSPF or via an external Path Computation Engine (PCE).

TE policy tunnel

TE policy tunnels are source-routed traffic engineered end-to-end segment routing paths, where routing constraints such as strict or loose hops can be used to determine a data path through a network. TE-policy tunnels can have one (uncolored) or more (colored) segment-lists that are active at any given time. These segment-lists follow the guidelines set by an uncolored SR-MPLS TE-Policy. Note that in 7x50 SR OS routers, uncolored SR-MPLS TE-Policy segment lists are referred to as SR-TE LSPs.

Note: The TTM preference value for the uncolored TE policy tunnel is set to eight to align with the 7x50 SR OS router SR-TE LSP set. For information about the TTM preferences for all supported tunnel types on 7250 IXR and 7730 SXR systems, see the "TTM preferences for supported tunnel types" section in the *SR Linux MPLS Guide*.

Uncolored SR-MPLS TE Policy segment lists are similar to traditional traffic-engineered LSPs such as RSVP-TE and offer a natural migration path, except that uncolored SR-MPLS TE Policy segment lists do not have a mid-point state. Each intermediate and tail-end router is unaware of the segment list's presence because no signaling protocol is used to create the path. The path can be computed locally by the ingress PE or by offloading the path computation to an external controller (PCE).

11.1.2 Binding segments

Binding Segment, or BSID, is an SID value that opaquely represents an SR-MPLS TE policy to upstream routers. When a packet with a BSID as the top label is received, the router selects an SR-MPLS TE policy based on the BSID to route the packet. BSIDs provide isolation or decoupling between different source-routed domains and improve overall network scalability.

A BSID can be assigned to a uncolored SR-MPLS TE policy, which can have multiple segment lists (primary and secondary). The BSID remains the same, but the actual path taken can change based on the availability and preference of the segment list.

The SR-MPLS TE policy switches to the secondary segment list if the primary segment list becomes unavailable because of link or node failure. This ensures traffic continues to be routed without interruption.

Suppose a network has four nodes - N1, N2, N3, and N4. The segment list and BSID are configured as follows:

- primary segment list has four SIDs: 1,2,3,4
- secondary segment list has two SIDs : 1,4
- BSID: 16001

When a data packet with BSID 16001 is received, the ingress router applies the SR-MPLS TE policy associated with BSID 16001. This policy includes an explicit path consisting of a sequence of nodes (segments) the packet must traverse. All SIDs listed in a specific segment list are included in the packet header.

When a packet is sent with a primary segment list (1, 2, 3, 4), the packet traverses through nodes in the order of these segment IDs. If the primary segment list is unavailable or needs rerouting, the packet can use the secondary segment list (1, 4) instead. This secondary segment list serves as a backup path for the packet.

11.1.3 Configuring SR-MPLS TE policy

Procedure

SR Linux supports default network instance TE policies for configuration of traffic engineered tunnels.

To configure SR-MPLS TE policy, perform the following tasks:

1. Enable TE database in IGP (IS-IS) instance.
2. Configure a static label range.
3. Assign the static MPLS label range to the BSID.
4. Define explicit path.
5. Configure the TE features.
6. Enable advertisement of IS-IS TE TLVs/sub-TLVs.
7. Configure the SR-MPLS TE policy using the defined explicit path and BSID.

Example: Enable TE

Use the command `network-instance* [name] protocols isis!+ instance* [name] te-database-install!` to enable TE. After TE is enabled, the IS-IS instance installs topology and TE information into the TE database.

```
--{ + running }--[  ]--
# info with-context network-instance default protocols isis instance 0
    network-instance default {
        protocols {
            isis {
                instance 0 {
                    te-database-install {
                    }
                }
            }
        }
    }
```

Example: Configuring a static label range

Use the command, `.system.mpls.label-ranges.` to configure the static label range. The static label range is used by all participating routers within the SR domain. In the configuration example, a static MPLS range (`srl-static-label-range`) is set from 16001-104857.

```
--{ + running }--[  ]--
# info with-context system mpls label-ranges static srl-static-label-range
    system {
        mpls {
            label-ranges {
                static srl-static-label-range {
                    shared false
                    start-label 16001
                    end-label 104857
                }
            }
        }
    }
```

Example: Assigning a static MPLS range to the BSID

The following example configures the BSID in SR-capable routers to align with the configured label range (`srl-static-label-range`).

```
--{ + running }--[  ]--
# info with-context network-instance default traffic-engineering-policies binding-sid
    network-instance default {
        traffic-engineering-policies {
            binding-sid {
                static-label-block srl-static-label-range
            }
        }
    }
```

Example: Define explicit path

Define the explicit path that the SR-MPLS TE policy follows. This involves configuring the intermediate hops (nodes or segments) that the traffic follows along the explicit path.

The following example configures an explicit path that can be used as a primary path (`e_prim_path`).

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies explicit-paths
 path e_prim_path
    network-instance default {
        traffic-engineering-policies {
            explicit-paths {
                path e_prim_path {
                    hop 10 {
                        ip {
                            ip-address 10.0.0.2
                            hop-type strict
                        }
                    }
                    hop 20 {
                        ip {
                            ip-address 10.0.0.3
                            hop-type strict
                        }
                    }
                    hop 30 {
                        ip {
                            ip-address 10.0.0.4
                            hop-type strict
                        }
                    }
                }
            }
        }
    }
```

The following example configures another explicit path that can be used as a secondary path (`e_sec_path`).

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies explicit-paths
 path e_sec_path
    network-instance default {
        traffic-engineering-policies {
            explicit-paths {
                path e_sec_path {
                    hop 40 {
                        ip {
                            ip-address 10.0.0.5
                            hop-type loose
                        }
                    }
                }
            }
        }
    }
```

Example: Configuring the TE link attributes

For information about configuring the TE attributes in the SR domain, see Traffic Engineering.

Example: Enabling advertisement of IS-IS TE TLVs/sub-TLVs

For information about enabling advertisement of IS-IS TE TLVs/sub-TLVs, see the "Enabling advertisement of IS-IS TE TLVs/sub-TLVs" section in the *SR Linux Routing Protocols Guide*.

Example: Configuring SR-MPLS TE policy

The following example configures the SR-MPLS TE policy using the defined explicit path and the BSID. The `re-optimization-timer` is set to 50 minutes, indicating that the router recomputes the path every 50 minutes.

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies policy srl_mpls_
te_policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_mpls_te_policy {
                policy-type sr-mpls-uncolored
                endpoint 10.0.0.4
                re-optimization-timer 50
                binding-sid {
                    mpls-label 16001
                }
                segment-list 10 {
                    explicit-path e_prim_path
                    segment-list-type primary
                }
                segment-list 20 {
                    explicit-path e_sec_path
                    segment-list-type secondary
                }
            }
        }
    }
```

Note: When a secondary segment list is up and forwarding and the standby segment list becomes available, the standby segment list becomes the new active segment list. This is because the standby segment list is preferred over the secondary segment list, and the secondary segment list is removed.

11.1.4 Configuring BSID for a TE policy

Procedure

You can configure a BSID to reference an entire set of segment lists (LSPs) of a TE policy. For more information about BSID, see Binding segments.

The high-level steps are as follows:

1. Configure a static MPLS label range.
2. Assign the static MPLS label range to a BSID.
3. Configure the BSID to reference a TE policy.

Example: Configure a static MPLS label range

The following example defines a static MPLS label range (`srl-static-label-range`) for use by the SRLB. The `start-label` or `end-label` value of the static LSP label range must be within the range 16-1048575.

```
--{ + running }--[  ]--
# info with-context system mpls label-ranges static srl-static-label-range
    system {
        mpls {
            label-ranges {
                static srl-static-label-range {
                    shared false
                    start-label 16001
                    end-label 104857
                }
            }
        }
    }
```

Example: Assign the static MPLS label range to the BSID

The following example configures the BSID in SR-capable routers to correspond to the configured label range (`srl-static-label-range`).

```
--{ + running }--[  ]--
# info with-context network-instance default traffic-engineering-policies
    network-instance default {
        traffic-engineering-policies {
            binding-sid {
                static-label-block srl-static-label-range
            }
        }
    }
```

Example: Configure the BSID to reference a TE policy

In the following example, the MPLS label value `16008` falls within the defined MPLS label range (`srl-static-label-range`) and serves as the BSID for the configured SR-MPLS TE policy (`srl_mpls_te_policy`).

```
--{ + running }--[  ]--
# info with-context network-instance default traffic-engineering-policies policy srl_mpls_
te_policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_mpls_te_policy {
                binding-sid {
                    mpls-label 16008
                }
            }
        }
    }
```

11.1.5 Displaying TE database information

Procedure

The following example displays TE database information.

Example

```
--{ + running }--[  ]--
# info from state with-context network-instance default traffic-engineering-policies
 policy-database
    network-instance base {
        traffic-engineering-policies {
            policy-database {
                sr-uncolored {
                    policy to-C-srte-ipv4-bsid-1 protocol-origin local {
                        endpoint 1.1.1.3
                        policy-type sr-mpls-uncolored
                        tunnel-id 21
                        metric 4000
                        created-time "2025-10-10T03:39:39.235Z (3 minutes ago)"
                        segment-list-count 1
                        active-segment-list-index 1
                        oper-state up
                        oper-state-change-count 1
                        last-oper-state-change "2025-10-10T03:39:39.494Z (3 minutes ago)"
                        binding-sid {
                            mpls-label 200007
                            allocation-status true
                        }
                        segment-list 1 {
                            oper-state up
                            forwarding-state active
                            last-oper-state-change "2025-10-10T03:39:39.494Z (3 minutes
 ago)"
                            oper-state-change-count 1
                            segment-list-type standby
                            segment-list-preference 100
                            explicit-path to-C-ip-Sys-BDFE-ipv4
                            metric 4000
                            igp-metric 4000
                            lsp-id 21
                            last-retry-attempt "2025-10-10T03:39:39.235Z (3 minutes ago)"
                            next-reoptimization-attempt "2025-10-10T04:09:39.235Z (26
 minutes from now)"
                            path-computation-requests 1
                            computed-segments {
                                segment 1 {
                                    hop-type ipv4
                                    ip-address 1.1.1.4
                                    is-loose true
                                    router-id 1.1.1.4
                                    sid-type node-sid
                                    sid-value {
                                        mpls-label 24000
                                    }
                                }
                                segment 2 {
                                    hop-type ipv4
                                    ip-address 1.1.1.6
                                    is-loose true
                                    router-id 1.1.1.6
```

```
                                sid-type node-sid
                                sid-value {
                                    mpls-label 26000
                                }
                            }
                            segment 3 {
                                hop-type ipv4
                                ip-address 1.1.1.5
                                is-loose true
                                router-id 1.1.1.5
                                sid-type node-sid
                                sid-value {
                                    mpls-label 25000
                                }
                            }
                            segment 4 {
                                hop-type ipv4
                                ip-address 1.1.1.3
                                is-loose true
                                router-id 1.1.1.3
                                sid-type node-sid
                                sid-value {
                                    mpls-label 23000
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

11.1.6 SR-MPLS TE policy path computation

When an SR-MPLS TE policy is configured on the router, the router can compute its path or an external TE controller. A TE policy can be based on an implicit or explicit path. The implicit path is the default mode and does not require configuration..

SR Linux supports the following path computation methods configurable individually for each TE policy segment list:

- Explicit path: allows the user to manually configure each TE Policy segment list path using an explicit list of SID values.
- Local Constrained Shortest Path First (CSPF): is the default path computation method, allowing routers within a network to calculate TE Policy segment list paths autonomously using CSPF algorithms. . Unlike explicit paths defined manually, local CSPF computation dynamically selects paths based on constraints and real-time network conditions.

  Note:
  Explicit paths are preferred, so the CSPF calculations are limited to considering only TE constraints within the scope of the configured hops.

- Path Computation Element Protocol (PCEP): allows the SR Linux router, acting as a PCE client (PCC), to forward requests for SR-MPLS TE path computations to the PCE.

11.1.7 Explicit path

The SR-MPLS TE policy path can be manually configured using an explicit list of SID values or nexthop IP addresses. Explicit path configuration is supported for primary, secondary, and standby paths.

When configuring an explicit path, you can specify either an MPLS label (`mpls-label`) or the IP address (`ip-address`) of the intermediary hops that the LSP needs to pass to the egress router. However, using an IP address requires IP-to-label translation to map the address to the corresponding SID in the SR network.

When using explicitly configured SIDs for a path that includes SIDs or IP address hops, you must provide all the necessary SIDs to reach the destination. The router verifies if the first SID provided is present in the router tunnel table.

When explicit SIDs are used in a path, the router prioritizes these SIDs over any configured path computation methods such as PCEP or CSPF at the TE policy segment list level. The router can establish the path based on the specified SIDs alone, disregarding the configured path computation for TE policy segment list.

When a TE-Policy (`sr-mpls-uncolored`) includes both SID-labeled paths and paths calculated using local-CSPF, the router cannot guarantee SRLG diversity between the CSPF and SID-labeled paths. This is because CSPF does not know about the SID-labeled paths, which are not included in the TE database.

Note:
Paths containing explicit SID values can only be used by SR-MPLS TE policy tunnels.

11.1.7.1 Configuring explicit-path segment lists

Procedure

When configuring an explicit path, specify either an MPLS label (`mpls-label`) or the IP address (`ip-address`) for each next hop in the path. The explicit path can include strict, loose, or a combination of both types of hops.

Example: Configuring an explicit path MPLS label

The following example configures an explicit path using the MPLS label (mpls-label) for the intermediary hops that the TE policy segment list needs to pass to reach the egress router. The value of the `mpls-label` must be within the range 16-1048575.

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies explicit-paths
 path srl-explicit_path
    network-instance default {
        traffic-engineering-policies {
            explicit-paths {
                path srl-explicit_path {
                    hop 50 {
                        mpls-label 16002
                    }
                    hop 60 {
                        mpls-label 16003
                    }
                }
            }
        }
```

```
    }
```

Example: Configuring an explicit path using an IPv4 address

The following example configures an explicit path consisting of IPv4 addresses of the intermediary hops that the TE policy segment list needs to pass to reach the egress router.

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies explicit-paths
 path srl-explicit_path
    network-instance default {
        traffic-engineering-policies {
            explicit-paths {
                path srl-explicit_path {
                    hop 10 {
                        ip {
                            ip-address 10.0.0.2
                            hop-type strict
                        }
                    }
                    hop 20 {
                        ip {
                            ip-address 10.0.0.3
                            hop-type strict
                        }
                    }
                    hop 30 {
                        ip {
                            ip-address 10.0.0.4
                            hop-type strict
                        }
                    }
                }
            }
        }
    }
```

11.1.8 Local CSPF (Constrained Shortest Path First) based path computation

TE policy path computation using local CSPF is applicable in single-level IS-IS instances or when the network is expanded into multiple IGP areas or instances, an external PCE is required.

TE policy segment list does not require each router to be TE enabled, and the links do not have to be TE links. As long as the routers at each end of the link are SR enabled, local CSPF can calculate an end-to-end path.

Full CSPF path computation on the head-end router (PCC) results in a full explicit path to the destination. The PCC calculates an end-to-end path and the following applies:

- The computed path is a full explicit TE path.
- Each link is represented by an adjacency SID.
- CSPF returns a label stack list of adjacency SIDs.

A TE policy segment list can be re-signaled when a timer expires, when an operator issues a command, or in case of IGP events.

Paths computed by local CSPF contain an adjacency SID for each link in the path and the stack may contain numerous labels. When a `network-instance traffic-engineering-policies policy`

segment-list dynamic path-algorithm local-cspf te-constraints.segment-depth value exceeds the calculated LSP label stack size or the maximum segment depth of a downstream router is lower than the calculated LSP label stack size, the label stack can be reduced. The label stack reduction capability can replace a series of adjacency SIDs with a node SID. For loose-hop path computation, node SIDs or a combination of node and adjacency SIDs can be used.

Local CSPF is supported on both primary and secondary standby paths of a TE policy segment list.

A TE-policy segment list can have its path computed by local CSPF based on the metric type configured within the network-instance* [name] traffic-engineering-policies! policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints context. The supported metric types include igp, te, and delay, with igp being the default. For configuration example of local CSPF computatation using delay metric, see Configuring local CSPF path computation using delay metric.

11.1.8.1 Local CSPF path computation and SR protected interfaces

When SR is enabled and IGP adjacency is established over a link, the router advertises an adjacency SID in the adjacency SID sub-TLV. When Loop-Free Alternate (LFA), Remote LFA (RLFA), or Topology-Independent LFA (TI-LFA) is enabled, protected adjacencies have the backup flag (B-flag) set in the adjacency SID sub-TLV. Each adjacency is available for protection when LFA, RLFA, or TI-LFA is enabled. It is possible to remove protection on a specific link by setting the parameter local-sr-protection within the network-instance.traffic-engineering-policies.policy.segment-list.dynamic.te-constraints to none.

Local CSPF path calculation can set up a path that:

- only includes protected adjacencies - Here, the parameter local-sr-protection within the network-instance traffic-engineering-policies policy segment-list dynamic path-algorithm local-cspf te-constraints is set to mandated value.
- only includes unprotected adjacencies - Here, the parameter local-sr-protection within the network-instance traffic-engineering-policies policy segment-list dynamic path-algorithm local-cspf te-constraints is set to none value.
- can include both protected and unprotected adjacencies - Here, the parameter local-sr-protection within the network-instance traffic-engineering-policies policy segment-list dynamic path-algorithm local-cspf te-constraints is set to preferred value. This is a default option.

11.1.8.2 Configuring CSPF for path computation

Procedure

To configure a TE policy segment list path computation request to be forwarded to a local router CSPF, set the parameter path-algorithm value within the network-instance default traffic-engineering-policies.policy.segment-list dynamic to local-cspf.

Example: Configuring CSPF for path computation

In the following configuration example, the `path-algorithm` of `srl_test_policy` is set to `local-cspf`, and the `re-optimization-timer` is set to 50 minutes. At each 50-minute interval, the router recomputes the path based on local CSPF behavior.

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy {
                policy-type sr-mpls-uncolored
                endpoint 10.0.0.2
                re-optimization-timer 50
                segment-list 3 {
                    segment-list-type primary
                    dynamic {
                        path-algorithm local-cspf
                    }
                }
                segment-list 6 {
                    segment-list-type standby
                    dynamic {
                        path-algorithm local-cspf
                    }
                }
                segment-list 8 {
                    segment-list-type secondary
                    dynamic {
                        path-algorithm local-cspf
                    }
                }
            }
        }
    }
```

11.1.8.3 Local CSPF path computation using delay metric

A TE-policy segment list can have its path computed by local CSPF based on the metric type configured within the `network-instance* [name] traffic-engineering-policies! policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints` context. The supported metric types include `igp`, `te`, and `delay`, with `igp` being the default.

A TE-policy segment list, used by real-time or delay-sensitive user and control traffic, can have its path computed by local CSPF based on the delay metric.

For configuration example of local CSPF computatation using delay metric, see Configuring local CSPF path computation using delay metric.

To use the delay metric, set the metric type (`metric-type`) to `delay` under the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints` context.

When the `delay` metric is configured, the node selects the path with the lowest end-to-end delay from the ingress Label Edge Router performing `local-cspf` computation to the egress label edge router (eLER).

The `delay` metric values reported by all subinterfaces along the path are accumulated and compared against the configured delay metric limit. The total delay of the calculated path must stay within this limit.

The default setting for `delay-metric-limit` parameter is `no-limit` and is configured under the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints delay-metric-limit` context.

The TE-policy segment list is only resignaled after a failure, timer expiration, or similar events and is not based on changes to the delay metric.

During path recalculation, the latest delay values from the subinterfaces are used. The delay measurements are relayed to the traffic engineering database (TEDB) based on the following sequence of events: TWAMP, STAMP, or similar tools measure delay and publish to the internal IP module, which relays it via IGP (ISIS in this case). A new delay value learnt by a PE is published to the TE database in a manner comparable to other attributes. SR-TE subsequently queries the TEDB to compute an optimal path.

11.1.8.3.1 Configuring local CSPF path computation using delay metric

Procedure

To configure `local-cspf`path computation using delay metric, you must, set the parameter `metric-type` value within the `network-instance* [name] traffic-engineering-policies! policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints` context to `delay`.

Example

In the following example, the SR-MPLS TE-policy (`srl_test_policy`) path is calculated with a delay constraint of 30 microseconds. The static delay configured on interfaces is used as the basis for delay-based path computation. For an example of a static delay configuration, see the "Configuring interface delay" section of the SR Linux Interfaces Guide.

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy {
                policy-type sr-mpls-uncolored
                endpoint 10.0.0.2
                segment-list 3 {
                    segment-list-type primary
                    dynamic {
                        path-algorithm local-cspf
                        te-constraints {
                            metric-type delay
                            delay-metric-limit 30
                        }
                    }
                }
            }
        }
    }
```

11.1.8.4 Configuring exclude-srlg TE constraint

Procedure

SR Linux supports Shared Risk Link Group (SRLG) for local CSPF-based path computation. SRLGs identify links that share common infrastructure where a failure could impact all the links even though they may be pointing to different NHLFEs. For example, an SRLG group could represent all links that use separate fiber strands but is carried in the same fiber conduit. If the conduit is accidentally cut, all the fiber links are cut, meaning all interfaces using these fiber links will fail.

When an **exclude-srlg** parameter is configured for a primary segment list, the standby or the secondary segment list, the segment list attempts to ignore the links that share the specified SRLGs during path computation. This ensures path disjointness between the primary path and the secondary paths.

Example: Configure exclude-srlg TE constraint

In the following configuration example, the policy `srl_te_policy` defines an uncolored TE policy with a primary segment list. The policy specifies a local CSPF path algorithm and excludes an SRLG named `gray` from the path calculation.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering-policies policy srl_mpls_
te_policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_mpls_te_policy {
                policy-type sr-mpls-uncolored
                endpoint 10.0.0.4
                binding-sid {
                    mpls-label 16001
                }
                segment-list 1 {
                    segment-list-type primary
                    dynamic {
                        te-constraints {
                            exclude-srlg [
                                black
                            ]
                        }
                    }
                }
            }
        }
    }
```

Note:
While **exclude-srlg** is configured for the primary segment list, the secondary SRLG is configured for a secondary segment list. When the **secondary-srlg** parameter is set to **true** for a secondary segment list, the secondary segment list avoids the SRLGs of the primary segment. For configuration example, see Configuring exclude-srlg TE constraint.

11.1.8.5 Configuring secondary SRLG TE-constraint

Procedure

When the **secondary-srlg** is configured in the secondary segment, it ensures path disjointness between the primary path and the secondary paths.

In the following configuration example, the policy `srl_te_policy` defines an uncolored TE policy with a secondary segment list. The policy specifies a local CSPF path algorithm and sets the **secondary-srlg** parameter to `true`, indicating that SRLGs of the primary segment are not considered in the CSPF path calculation.

Example: Configure secondary SRLG TE-constraint

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering-policies policy srl_te_
policy    policy-type sr-mpls-uncolored
    network-instance default {
        traffic-engineering-policies {
            policy srl_te_policy {
                policy-type sr-mpls-uncolored
                endpoint 10.0.0.2
                re-optimization-timer 50
                segment-list 3 {
                    segment-list-type primary
                    dynamic {
                        path-algorithm local-cspf
                        te-constraints {
                            metric-type delay
                            delay-metric-limit 70
                            exclude-srlg [
                                black
                            ]
                        }
                    }
                }
                segment-list 8 {
                    segment-list-type secondary
                    dynamic {
                        path-algorithm local-cspf
                        te-constraints {
                            secondary-srlg true
                        }
                    }
                }
            }
        }
    }
```

11.1.9 Path Computation Element Protocol (PCEP)

The PCEP is one of several protocols used to communicate between a Wide Area Network (WAN) Software Define Networking (SDN) controller and network elements. The Nokia WAN SDN Controller is known as the Network Services Platform (NSP).

The SR Linux node operates as the Path Computation Client (PCC), while the Nokia NSP (or another third-party PCE) serves as Path Computation Element (PCE). PCEs are used for path computation of TE policy segment list in SR-MPLS domain.

PCEP provides mechanisms for PCEs to perform path computations in response to PCC requests.

Each router (SR Linux node) acting as a PCC initiates a PCEP session with the PCE in its domain. The PCC uses PCEP to send a path computation request for one or more TE policy segment lists to the PCE. The PCE then replies with a set of computed paths if it finds one or more paths that satisfy the set of constraints.

SR Linux supports the following PCE and PCC capabilities:

- base PCEP implementation, in accordance with RFC 5440
- active and passive stateful PCE LSP update, in accordance with RFC 8231, *Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE*
- delegation of LSP control to PCE
- synchronization of the LSP database (LSP-DB) with network elements for PCE-controlled LSPs and network element-controlled LSPs
- support for PCC-initiated LSPs, in accordance with RFC 8231, *Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE*

11.1.9.1 Base implementation of PCE

The base implementation of PCE uses the PCEP extensions defined in RFC 5440.

The main functions of the PCEP are:

- establishing, maintaining, and closing PCEP sessions
- generating path computation requests using the PCReq message
- generating path computation replies using the PCRep message
- generating notification messages (PCNtf) by which the PCEP speaker can inform its peer about events, such as path request cancellation by the PCC or path computation cancellation by the PCE
- generating error messages (PCErr) by which the PCEP speaker can inform its peer about errors related to processing requests, message objects, or TLVs

The following table lists the base PCEP messages and objects.

Table 8: Base PCEP message objects and TLVs

TLV, object, or message	Contained in object	Contained in message
OPEN object	—	OPEN, PCErr
Request Parameter (RP) object	—	PCReq, PCRep, PCErr, PCNtf
NO-PATH object	—	PCRep
END-POINTS object	—	PCReq

TLV, object, or message	Contained in object	Contained in message
METRIC object	—	PCReq, PCRep, PCRpt, PCInitiate
Explicit Route Object (ERO)	—	PCRep
Reported Route Object (RRO)	—	PCReq
LSPA object	—	PCReq, PCRep, PCRpt, PCInitiate
NOTIFICATION object	—	PCNtf
PCEP-ERROR object	—	PCErr
CLOSE object	—	CLOSE

The base (stateless) PCE computes paths without considering the existing state of network LSPs.

The behavior and limitations of the implementation of the objects in the preceding table are as follows:

- The PCE treats all supported objects received in a PCReq message as mandatory, regardless of whether the P-flag in the common header of the object is set (mandatory object) or not (optional object).
- The PCC implementation always sets the B-flag (B=1) in the METRIC object containing the hop metric value, which means that a bound value must be included in the PCReq message. The PCE returns the computed value in the PCRep message with flags set identically to the PCReq message.
- The PCC implementation always sets flags B=0 and C=1 in the METRIC object for the IGP or TE metric values in the PCReq message. This means that the request is to optimize (minimize) the metric without providing a bound. The PCE returns the computed value in PCRep message with flags set identically to the PCReq message.
- The IRO and LOAD-BALANCING objects are not supported in the NSP PCE feature. If the PCE receives a PCReq message with one or more of these objects, it ignores them regardless of the setting of the P-flag and processes the path computations normally.
- LSP path setup and hold priorities are configurable during SR-TE LSP configuration on the router, and the PCC passes the configurations on in an LSPA object. However, the PCE does not implement LSP preemption.
- The LSPA and METRIC objects are also included in the PCRpt message.

The following features are not supported in SR Linux:

- PCE discovery using IS-IS (as defined in RFC 5089) and OSPF (as defined in RFC 5088) along with corresponding extensions for discovering stateful PCE (as defined in *draft-sivabalan-pce-disco-stateful*)
- PCEP synchronization optimization (as defined in RFC 8232)
- jitter, latency, or packet loss link metric signaling in the PCE METRIC object (as defined in RFC 8233)

11.1.9.2 PCEP session establishment and maintenance

The PCEP protocol operates over TCP using the destination TCP port 4189. The PCE client (PCC) always initiates the connection. After the user configures the PCEP local IPv4 or IPv6 address and the peer IPv4 or IPv6 address on the PCC, the PCC initiates a TCP connection to the PCE. If both a local IPv4 and a

local IPv6 address are configured, the connection uses the local address of the same family as the peer address. When the connection is established, the PCC and PCE exchange OPEN messages, and this process initializes the PCEP session and exchanges the session parameters to be negotiated.

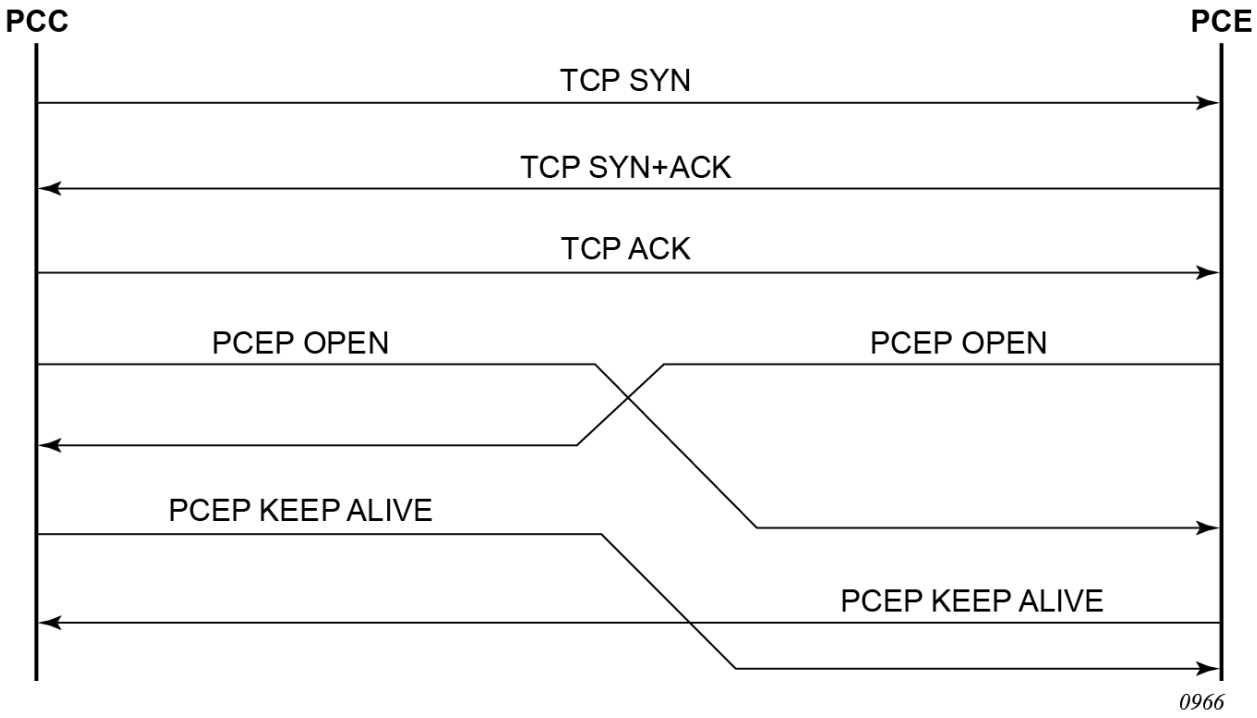
By default, the PCC attempts to reach the remote PCE address through the default network instance. PCE connectivity can be via default or management network instance. SR Linux node in the PCC role attempts an in-band connection to PCE via default `network.instance` which can be changed to management `network.instance` for out-of-band connectivity. The local IPv4 or IPv6 address configured by the user is always used to set up the PCEP session.

A keepalive mechanism is used as an acknowledgment of the acceptance of the session within the negotiated parameters. It is also used as a maintenance function to detect whether the PCEP peer is still alive.

The negotiated parameters include the keepalive timer and the dead timer and one or more PCEP capabilities, such as support for stateful PCE and the SR-MPLS TE LSP path type.

The following figure shows PCEP session initialization steps.

Figure 6: PCEP session initialization



Suppose the session to the PCE times out, the router acting as a PCC keeps the last successfully programmed path provided by the PCE until the session is re-established. Any subsequent TE policy segment list state change is synchronized when the session is re-established.

When a PCEP session with a peer time out or closes, the rate at which the PCEP speaker attempts to establish the session is subject to an exponential back-off mechanism.

11.1.9.3 PCEP over TLS

For Path Computation Element Communication Protocol (PCEP) security, SR Linux supports the usage of Transport Layer Security (TLS) as described in RFC 8253. PCEP provides mechanisms for PCEs to perform path computations in response to PCC requests.

In TLS mode, both the PCC and PCE must provide certificates for authentication. The PCE provides the server certificate to the PCC and requires the client certificate to authenticate the PCC.

PCEP over TLS (PCEPS) is secured using TLS on port 4189. The PCC is configured with a TLS client profile to initiate the TLS handshake. The PCE is configured with a TLS server profile to allow PCEPS.

When a TLS server profile is configured on the PCE, the PCE can establish TLS and non-TLS connections, in PCE secured (PCES) and PCE modes.

According to RFC 8253, the PCC and PCE can negotiate their TLS capability and decide whether to establish a connection. However, SR Linux supports only PCC strict mode, meaning if TLS is configured on the PCC, it only establishes a connection if the PCE supports PCEPS.

PCC behavior

On the PCC, SR Linux supports only strict TLS. That is, both PCE and PCC must support TLS and perform a successful TLS handshake before the TLS wait timer expires. Otherwise, the PCC retries the connection after 60 seconds.

11.1.9.3.1 TLS handshake

SR Linux uses TLS 1.2 or a later version, with the PCC acting as the TLS client. For authentication, certificate mutual authentication is employed, where the PCC sends its certificate to the PCE. The following cipher suites are supported as described in RFC 6460:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

For peer authentication, the configuration supports X.509 certificates.

The following figure shows the TLS handshake.

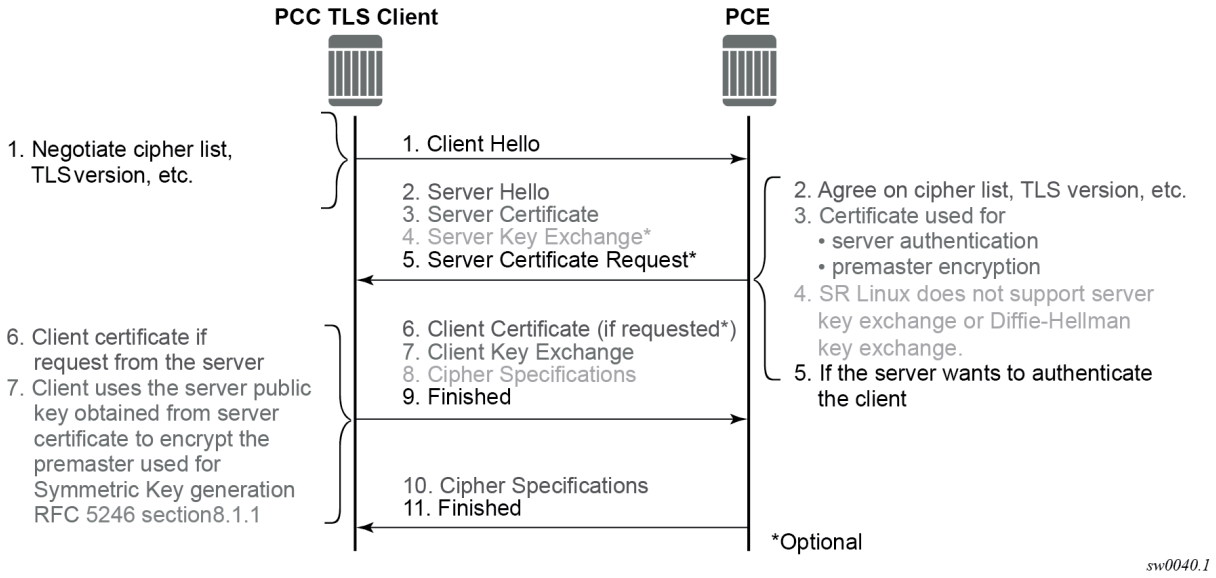


Table 9: TLS handshake step descriptions

Step	Description
1	The TLS handshake begins with the PCC TLS Client (SR Linux) Hello message. This message includes the cipher list that the client wants to use and negotiate, among other information.
2	The server sends back a Server Hello message, along with the first common cipher found on both the client and server cipher lists. The common cipher is used for data encryption.
3	The server sends a server certificate message, in which the server provides a certificate that the client can use to authenticate the server identity. The public key of this certificate (RSA key) can also be used to encrypt the symmetric key seed used by the client and server to create the symmetric encryption key. This occurs only if the PKI is using RSA for asymmetric encryption.
4	SR Linux does not support server key exchange or Diffie-Hellman key exchange.
5	The server can optionally be configured to request a certificate from the client to authenticate the client.
6	If the server requests a client certificate, the client must provide it by using a client certificate message. If the client does not respond to the certificate request, the server drops the TLS session.
7	The client uses the public server asymmetric key included in the server certificate to encrypt a seed. The client and server use this seed to create the identical symmetric key used for encrypting and decrypting data plane traffic.
8	The client sends a cipher spec message to switch encryption to this symmetric key.
9	The client successfully finishes the handshake.
10	The server sends a cipher spec message to switch encryption to this symmetric key.
11	The server successfully finishes the handshake.

11.1.9.3.2 PCEP session over TLS

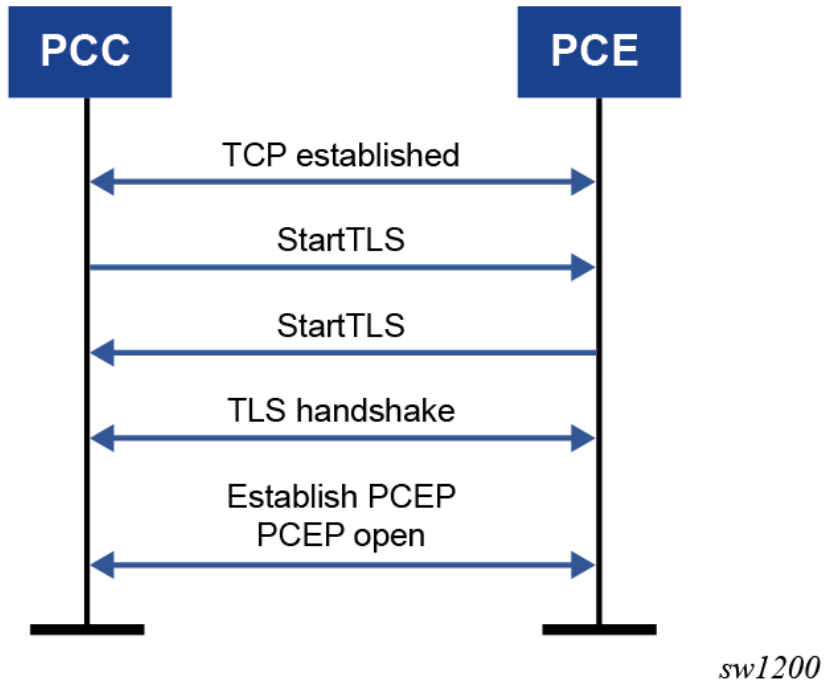
To establish a PCEP session over TLS as specified in RFC 8253, the PCC sends a StartTLS message to the PCE to initiate the TLS negotiation. The PCC activates the StartTLS timer and waits for the StartTLS message from the PCE.

You can configure the timer using the `network-instance.protocols.pcep.pcc.peer.tls-wait-timer` command. The default timer is 60 seconds.

If the PCE is TLS-capable and sends back a StartTLS message before the StartTLS timer expires, the TLS handshake is initiated. If the PCE sends an Open message or does not send back a StartTLS message, the PCC responds with an error message, closes the TCP connection, and then attempts to re-establish the connection. The PCEP Message-Type field of the PCEP common header for the StartTLS message is set to 13, as specified in RFC 8253.

The following figure shows the establishment of a PCEP session over TLS.

Figure 7: PCEP session over TLS



11.1.9.3.3 Configuring PCC with TLS client profile and TLS wait timer

Procedure

The PCC is set up with a TLS client profile, which enables it to initiate the TLS handshake process. The PCE is configured with a TLS server profile, allowing it to support PCEP communications over a TLS-secured connection.

Use the command **network-instance.protocols.pcep.pcc.peer.tls-client-profile** to configure a PCC TLS profile.

To configure the timer, use the **network-instance.protocols.pcep.pcc.peer.tls-wait-timer** command. The default timer is 60 seconds.

Example: Configuring PCC with TLS client profile and TLS wait timer

In the following configuration example, a PCC TLS profile (`clab-profile`) is configured. The PCC activates the StartTLS timer of 80 seconds and waits for the StartTLS message from the PCE.

```
--{ * candidate shared default }--[  ]--
# info with-context network-instance default protocols pcep pcc
    network-instance default {
        protocols {
            pcep {
                pcc {
                    peer 192.168.0.5 {
                        tls-client-profile clab-profile
                        tls-wait-timer 80
                    }
                }
            }
        }
    }
```

11.1.9.4 PCEP parameters

Procedure

SR Linux supports configuring of PCEP parameters at the `.network-instance .protocols.` level. On the PCE, the configured parameter values are used on sessions to all PCCs.

- `keepalive`
- `dead-timer`
- `unknown-message-rate`
- `redelegation-timer`
- `state-timer`
- `report-path-constraints`
- `peer`

Example: Configuring keepalive timer

A PCEP speaker (PCC or PCE) must send a keepalive message if no other PCEP message is sent to the peer at the expiry of this timer. This timer is restarted every time a PCEP message is sent or the keepalive message is sent.

The keepalive mechanism is asymmetric, meaning that each peer can use a different `keepalive` timer value.

The range of this parameter is 1 to 255 seconds, and the default value is 30 seconds. If the `keepalive` value is not explicitly configured, the system defaults to the predefined default value.

Note:
The `keepalive` parameter cannot be modified while the PCEP session is operational.

```
--{ candidate shared default }--[  ]--
```

```
info with-context network-instance default protocols pcep pcc keepalive
    network-instance default {
        protocols {
            pcep {
                pcc {
                    keepalive 254
                }
            }
        }
    }
```

Example: Configuring dead-timer

This timer tracks the amount of time a PCEP speaker (PCC or PCE) waits after the receipt of the last PCEP message before declaring its peer down.

The dead timer mechanism is asymmetric, meaning that each PCEP speaker can propose a different dead timer value to its peer to use to detect session timeouts.

The range of this parameter is 1 to 255 seconds, and the default value is 120 seconds. If the `dead-timer` value is not explicitly configured, the system defaults to the predefined default value.

Note:
The `dead-timer` parameter cannot be modified while the PCEP session is operational.

```
--{ candidate shared default }--[  ]--
info with-context network-instance default protocols pcep pcc dead-timer
    network-instance default {
        protocols {
            pcep {
                pcc {
                    dead-timer 50
                }
            }
        }
    }
```

Example: Configuring unknown message rate

You can configure the maximum rate of unknown messages which can be received on a PCEP session. When the rate of received unrecognized or unknown messages reaches this limit, the PCEP speaker closes the session to the peer.

Note:
The `unknown-message-rate` parameter can be modified while the PCEP session is operational.

```
--{ candidate shared default }--[  ]--
 info with-context network-instance default protocols pcep pcc unknown-message-rate
    network-instance default {
        protocols {
            pcep {
                pcc {
                    unknown-message-rate 65
                }
            }
        }
    }
```

Example: Configuring session redelegation timer

Redelegation timers (known as the Redelegation Timeout Interval in RFC 8231) are started when the PCEP session goes down or the PCE signals overload.

Note: The redelegation timer applies only to PCE-initiated TE policy segment lists.

If the PCEP session to the PCE goes down, all delegated PCC-initiated LSPs have their state maintained in the PCC and are not timed out. The PCC continues to attempt to re-establish the PCEP session. When the PCEP session is re-established, the LSP database is synchronized with the PCE, and any LSP that went down since the last time the PCEP session was up has its path updated by the PCE.

The range of the parameter`redelegation-timer` is 1 to 3600 seconds, and the default value is 90 seconds. If the `redelegation-timer` value is not explicitly configured, the system defaults to the predefined default value.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols pcep pcc redelegation-timer
    network-instance default {
        protocols {
            pcep {
                pcc {
                    redelegation-timer 2500
                }
            }
        }
    }
```

Example: Configuring state timers

State timers (known as the State Timeout Interval in RFC 8231) are started when the PCEP session goes down or the PCE signals overload.The state timer dictates how long the state should be maintained once the PCE connection is lost and must be greater than the redelegation timer.

Note: The state timer applies only to PCE-initiated TE policy segment lists.

```
--{ candidate shared default }--[  ]--
info with-context network-instance default protocols pcep
    network-instance default {
        protocols {
            pcep {
                pcc {
                    state-timer {
                        timer 30
                        timer-action none
                    }
                }
            }
        }
    }
```

Example: Configuring report path constraints

Use the parameter `report-path-constraints` to enable or disable the path constraints in PCC report. When set to `false`, the TE policy segment list path constraints are not included in the report message that the PCC sends to the PCE.

Note:
The `report-path-constraints` parameter can be modified while the PCEP session is operational.

```
--{ candidate shared default }--[  ]--
info with-context network-instance default protocols pcep pcc report-path-constraints
    network-instance default {
        protocols {
            pcep {
                pcc {
                    report-path-constraints true
                }
            }
        }
    }
```

Example: Configuring peer PCEP IP address

You can configure the IP address of a peer PCEP speaker. The address is used as the destination address in the PCEP session messages to a PCEP peer.

The `preference` parameter allows the PCC to select the preferred PCE when both have their PCEP sessions successfully established. A maximum of two PCEP peers is supported.

The PCE peer that is not in overload is always selected by the PCC as the active PCE. However, if neither of the PCEs are signaling the overload state, the PCE with the higher numerical preference value is selected, and in case of a tie, the PCE with the lower IP address is selected. To change the value of the `preference` parameter, the peer must be deleted and recreated.

Note:
The `peer` parameter cannot be modified while the PCEP session is operational.

In the following example two PCEP peers are configured.

```
--{ candidate shared default }--[  ]--
info with-context network-instance default protocols pcep pcc
    network-instance default {
        protocols {
            pcep {
                pcc {
                    unknown-message-rate 65
                    report-path-constraints true
                    dead-timer 89
                    redelegation-timer 2500
                    state-timer {
                        timer 30
                        timer-action none
                    }
                    peer 192.168.0.5 {
                        preference 5
                    }
                    peer 192.168.0.10 {
                        preference 1
```

```
                    }
                }
            }
        }
    }
```

11.1.9.5 PCE connection

Procedure

By default, the PCC attempts to reach the remote PCE address through the default network instance. PCE connectivity can be via default or management network instance. SR Linux node in the PCC role attempts an in-band connection to PCE via default `network.instance` which can be changed to management `network.instance` for out-of-band connectivity. The local IPv4 or IPv6 address configured by the user is always used to set up the PCEP session.

Example: Configuring an in-band reachability

In the following example, PCC is configured to connect to the PCE server with an in-band reachability.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols pcep pcc peer 193.0.0.1
    network-instance default {
        protocols {
            pcep {
                pcc {
                    peer 193.0.0.1 {
                        admin-state enable
                        local-address 192.0.0.5
                        preference 1
                        network-instance default
                    }
                }
            }
        }
    }
```

Example: Configuring an out-of-band reachability

In the following example, PCC is configured to connect to the PCE server with an out-of-band reachability.

```
--{ * candidate shared default }--[  ]--
info with-context network-instance default protocols pcep pcc peer 193.0.0.1
    network-instance default {
        protocols {
            pcep {
                pcc {
                    peer 193.0.0.1 {
                        admin-state enable
                        local-address 192.0.0.5
                        preference 1
                        network-instance mgmt
                    }
                }
            }
        }
```

```
    }
```

11.1.9.6 Stateful PCE

A stateful PCE implementation maintains an up-to-date Traffic Engineering Database (TED) and LSP State Database (LSP-DB).

The TED includes the network topology and resource state. The LSP-DB stores attributes of all active LSPs in the network, such as their paths through the network, bandwidth usage, switching types, and TE-policy constraints. This state information computes constrained paths while considering individual segments and their interdependency.

The primary function of stateful PCE, as opposed to the base PCE implementation, is the ability to synchronize the TE-Policy (`sr-mpls-uncolored`) segment list information between the PCC and the PCE. This function allows the PCE to have all the required segment list information to optimize and update the segment list paths.

The following table describes the messages and objects supported by stateful PCE in SR Linux.

Table 10: PCEP stateful PCE extension objects and TLVs

TLV, object, or message	Contained in object	Contained in message
Path Computation State Report (PCRpt) message	—	New message
Path Computation Update Request (PCUpd) message	—	New message
Stateful PCE Capability TLV	OPEN	OPEN
Stateful Request Parameter (SRP) object	—	PCRpt, PCErr, PCInitiate
LSP object	ERO	PCRpt, PCReq, PCRep, PCInitiate
LSP Identifiers TLV	LSP	PCRpt
Symbolic Path Name TLV	LSP, SRP	PCRpt, PCInitiate
LSP Error Code TLV	LSP	PCRpt

The following behavior and limitations apply to the implementation of the objects listed in the preceding table:

- PCC and PCE support all PCEP capability TLVs defined in this document and always advertise them. If the OPEN object received from a PCEP speaker does not contain one or more of the capabilities, neither the PCE nor the PCC use them during the specific PCEP session.
- The PCC always includes the LSP object in the PCReq message to ensure that the PCE can correlate the PLSP ID for this LSP when a subsequent PCRpt message arrives with the delegation bit set. The PCE, however, still honors a PCReq message without the LSP object.

- PCE path computation only considers the bandwidth used by LSPs in its LSP-DB. As a result, in the following situations, the PCE path computation does not accurately account for the bandwidth used in the network:
  - when there are LSPs that are signaled by the routers but are not synchronized with the PCE. The user can enable the reporting of the LSP to the PCE LSP database for each LSP.
  - when the stateful PCE is peering with a third-party stateless PCC, implementing only the original RFC 5440. While the PCE is able to bring the PCEP session up, the LSP database is not updated, because stateless PCC does not support the PCRpt message. As such, PCE path computation does not accurately account for the bandwidth used by these LSPs in the network.
- The PCE ignores the reoptimize flag (R-flag) in the PCReq message when acting in stateful-passive mode for a specific LSP and always returns the new computed path, regardless of whether it is link-by-link identical or has the same metric as the current path. The PCC decides whether to initiate the new path in the network.
- The SVEC object is not supported in SR Linux . If the PCE receives a PCReq message with the SVEC object, it ignores the SVEC object and treats each path computation request in the PCReq message as independent, regardless of the setting of the P-flag in the SVEC object common header.
- When an LSP is delegated to the PCE, there can be no prior state in the NRC-P LSP database for the LSP. This could be because the PCE did not receive a PCReq message for the same PLSP-ID. For the PCE to become aware of the original constraints of the LSP, the following additional procedures are performed:
  - The PCC appends a duplicate of each of the LSPA and METRIC objects in the PCRpt message. The only difference between the two objects of the same type is that the P-flag is set in the common header of the duplicate object to indicate a mandatory object for processing by the PCE.
  - The value of the metric in the duplicate object contains the original constraint value, while the first object contains the operational value. This is applicable to hop metrics in the METRIC object only. The SR Linux PCC does not support putting a bound on the IGP or TE metric in the path computation.
  - The path computation on the PCE uses the first set of objects when updating a path, if the PCRpt message contains a single set. If the PCRpt message contains a duplicate set, the PCE path computation must use the constraints in the duplicate set.
  - For interoperability, implementations compliant to PCEP standards accept the first metric object and ignore the second object without additional error handling. Because there are LSPA objects, the **report-path-constraints** command is provided in the PCC on a per-PCEP session basis to disable the inclusion of the duplicate objects. Duplicate objects are included by default.

Active stateful PCE

An active stateful PCE updates TE-Policy (`sr-mpls-uncolored`) segment list information for those TE-Policies whose PCCs have delegated control to the PCE. When you enable PCE control for one or more TE-Policy (`sr-mpls-uncolored`), the PCE takes over the path, updating and periodically re-optimizing the TE-policy.

Passive stateful PCE

A passive stateful PCE uses TE-Policy (`sr-mpls-uncolored`) segment list information to optimize path computation but does not actively modify the segment list state. In this role, the PCE considers other segment lists on the router, taking into account their resource usage while computing paths for the segment lists it controls, but it does not directly update the segment list state itself.

11.1.9.7 TE policy segment list path initiation

A TE policy segment list that is configured on the router is called a PCC-initiated TE policy segment list.

SR Linux supports the following modes of operation configurable per TE policy segment list.

- PCC-initiated and PCC-controlled TE policy segment list: TE policy segment list path is computed and updated by the router acting as a PCC.
- PCC-initiated and PCE-computed TE policy segment list: TE policy segment list path is computed by the PCE at the request of the PCC.

The configured `net-instance.traffic-engineering.te-router-id` is the source IP address in PCEP requests or reports. The network operator must ensure that the configured IP address is advertised via IGP.

In scenarios where `net-instance.traffic-engineering.te-router-id` is not configured, the PCE computed or PCE controlled segment list becomes inactive.

11.1.9.7.1 PCC-initiated and PCE-computed TE policy segment list

In this mode of operation, the TE policy segment list path is computed by the PCE at the request of the PCC. The PCE calculates the route of the TE policy segment list as requested by the PCC using the command, `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] dynamic! path-algorithm?`.

A PCC-initiated and PCE-computed TE policy segment list supports the passive stateful mode. This allows the PCE to perform path computation only at the request of the PCC; the PCC retains control. The capability exists to report and synchronize the TE policy segment list information with the LSP database of a stateful PCE server using the command `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] pce-report?`.

11.1.9.7.2 Configuring PCC-initiated and PCC-controlled TE policy segment list

Procedure

To authorize the PCC to compute the paths for the TE policy segment list, set the parameter `pce-control` value within the `network-instance default traffic-engineering-policies.policy.segment-list` to `false`.

Example: Configuring TE policy segment list path computation by PCC

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy {
                endpoint 192.168.2.1
                segment-list 3 {
                    pce-control false
                    pce-report true
                }
                segment-list 13 {
```

```
                    pce-control false
                    pce-report true
                }
            }
        }
    }
```

11.1.9.7.3 PCC-initiated and PCC-controlled TE policy segment list

When the TE policy segment list path is computed and updated by the router acting as a PCE Client (PCC), the TE policy segment list is called a PCC-initiated and PCC-controlled TE policy segment list. A PCC-initiated and PCC-controlled TE policy segment list has the following characteristics:

- can contain strict or loose hops, or a combination of both
- supports both a basic hop-to-label translation and a full CSPF as a path computation method
- capability exists to report and synchronize the TE policy segment list information with the LSP database of a stateful PCE server using the command `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] pce-report?` . The PCE cannot update the TE policy segment list path; the PCC maintains control of the TE policy segment list.

11.1.9.7.4 Configuring PCC-initiated, PCE-computed and Controlled TE policy segment lists

Procedure

To authorize the PCE to control the path and delegate the path it computes, set the parameter `pce-control` value within the `network-instance default traffic-engineering-policies.policy.segment-list` to `true`.

Example: Configuring TE policy segment list path computation and control by PCE

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy sr-mpls-te-policy
    network-instance default {
        traffic-engineering-policies {
            policy sr-mpls-te-policy {
                policy-type sr-mpls-uncolored
                segment-list 4 {
                    pce-control true
                }
            }
        }
    }
```

11.1.9.7.5 Configuring PCC-initiated and PCE-computed TE policy segment list

Procedure

To delegate path computation for the TE policy segment list to the PCE, set the parameter `path-algorithm` value within the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] dynamic!` to `pce`.

Example: Configuring TE policy segment list path computation by PCE

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy {
                endpoint 192.168.2.1
                segment-list 10 {
                    segment-list-type primary
                    pce-report true
                    dynamic {
                        path-algorithm pce
                    }
                }
            }
        }
    }
```

Example: Configuring a TE policy segment list to synchronize the LSP database with a stateful PCE

To report a TE policy segment list and synchronize the LSP database of a stateful PCE server, set the parameter `pce-report` value within the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index]` to `true`.

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl-mpls-
te-policy
    network-instance default {
        traffic-engineering-policies {
            policy srl-mpls-te-policy {
                policy-type sr-mpls-uncolored
                segment-list 4 {
                    pce-report true
                    dynamic {
                        path-algorithm pce
                    }
                }
            }
        }
    }
```

11.1.9.8 PCEP path computation using delay metric

A TE policy segment list, which is used by real-time or delay-sensitive user or control traffic, can make use of delay metric constraint for path computation. By setting the delay metric as upper bound, only paths whose total delay is less than or equal to the given threshold are considered viable during path selection.

To use the delay metric, set the metric type (`metric-type`) to `delay` under the `network-instance <default-instance>.traffic-engineering-policies policy <policy-name>.segment-list <segment-list-index>.dynamic.te-constraints` context.

When the `delay` metric is configured, the node selects the path with the lowest end-to-end delay from the ingress Label Edge Router to the egress router (eLER) based on the computed paths received through a PCC-ini setup.

The `delay` metric values reported by all subinterfaces along the path are accumulated and compared against the configured delay metric limit. The total delay of the calculated path must stay within this limit. The default setting for the `delay-metric-limit` parameter is `no-limit` and is configured under the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints delay-metric-limit` context.

The TE-policy segment list is only re-signaled in response to failures, timer expirations, or other similar events. Note that changes to the delay metric do not trigger re-signaling.

During path recalculation, the latest delay values from the subinterfaces are used. The delay measurements are relayed to the traffic engineering database (TEDB) based on the following sequence of events: TWAMP, STAMP, or similar tools measure delay and publish the results to IGP (IS-IS in this case), which relays the information. When the PE receives a new delay value, it updates its TE database, which is subsequently queried by the TE-Policy module to compute the optimal path.

11.1.9.9 Configuring PCEP path computation using delay metric

Procedure

To configure PCEP path computation using delay metric, you must set the `metric-type` parameter to `delay` in the `network-instance* [name] traffic-engineering-policies! policy* [policy-name] segment-list* [segment-list-index] dynamic! te-constraints` context.

Example: Configuring PCC-initiated, PCC-controlled TE policy segment list with delay metric upper bound

The static delay configured on interfaces is used as the basis for delay-based path computation. For an example of a static delay configuration, see the "Configuring interface delay" section of the SR Linux Interfaces Guide. In the following example, the SR-MPLS TE-policy (`srl_test_policy_pcc_with_upperbound`) path is configured with a delay metric upper bound of 30 microseconds, allowing only paths that meet this delay constraint to be considered during path computation.

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy_pcc_with_upperbound
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy_pcc_with_upperbound {
                endpoint 192.168.2.1
                segment-list 5 {
                    pce-control false
                    pce-report true
                    dynamic {
                        te-constraints {
                            metric-type delay
                            delay-metric-limit 30
                        }
                    }
                }
            }
        }
```

```
    }
```

Example: Configuring PCC-initiated, PCC-controlled TE policy segment list without delay metric upper bound

In the following example, the SR-MPLS TE-policy (`srl_test_policy_pcc_no_upperbound`) path is configured without a delay metric upper bound, allowing the path calculation to consider all possible paths are allowed.

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy_pcc_no_upperbound
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy_pcc_no_upperbound {
                endpoint 192.168.2.19
                segment-list 8 {
                    pce-control false
                    pce-report true
                    dynamic {
                        te-constraints {
                            metric-type delay
                            delay-metric-limit no-limit
                        }
                    }
                }
            }
        }
    }
```

Example: Configuring PCC-initiated, PCE-computed TE policy segment list with delay metric upper bound

In the following example, the SR-MPLS TE-policy (`srl_test_policy_pce_with_upperbound`) path is configured with a delay metric upper bound of 40 microseconds, allowing only paths that meet this delay constraint.

```
--{ +* candidate shared default }--[  ]--
info with-context network-instance default traffic-engineering-policies policy srl_test_
policy_pce_with_upperbound
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy_pce_with_upperbound {
                endpoint 192.168.9.1
                segment-list 8 {
                    pce-report true
                    dynamic {
                        path-algorithm pce
                        te-constraints {
                            metric-type delay
                            delay-metric-limit 40
                        }
                    }
                }
            }
        }
    }
```

11.1.9.10 Path computation fallback

The SR Linux router supports using a local path computation as a fallback for PCC-initiated SR-MPLS TE path computations. The fallback mechanism is triggered when the PCC notifies the MPLS process associated with the PCEP session that failed, or when the `network-instance test traffic-engineering-policies policy srl_mpls_te_policy re-optimization-timer` expires, and all configured TE-policy segment lists signal overload, preventing redelegation. However, the fallback mechanism is not triggered when the PCC is administratively down or has not yet been configured.

PCE-independent protection mechanisms such as LFA (Loop-Free Alternate) and ti-LFA (Topology-Independent LFA) continue to be supported. These protection mechanisms are managed by the node and do not rely on the PCE. Additionally, primary to standby TE-policy segment list path switching and vice versa remains supported. This means the node can switch between primary and standby paths even when the PCE is unavailable.

You can configure the fallback path algorithm by setting the parameter, **fallback-path-algorithm** in the `network-instance traffic-engineering-policies policy segment-list dynamic` context.

When the PCEP session recovers and at least one configured PCE does not have the overload bit set, the PCC notifies MPLS that the PCE is enabled, indicating that the PCC is ready to reuse the PCE for path computations.

SR Linux supports path computation at the segment list level. The following table illustrates the fallback computation behavior based on configuration of the PCE state, **path-algorithm**, and **fallback-path-algorithm** parameters.

Table 11: Fallback computation behavior based on the PCE state and path-algorithm configuration

Primary path algorithm (path-algorithm)	Secondary path algorithm (path-algorithm)	Fallback path algorithm method (fallback-path-algorithm)	PCE state	Behaviour
pce	**local-cspf**	Not configured	PCE is operational	Upon primary path failure, the secondary path is attempted based on the local CSPF computed path.
pce	**local-cspf**	**local-cspf**	PCE is in a down state	The primary path directly attempts a fallback computed path. Only if this fails is the secondary path attempted.
pce	**local-cspf**	**local-cspf**	PCE state is unknown (start-up phase)	The primary path first attempts the PCE-based path computation. If

Primary path algorithm (path-algorithm)	Secondary path algorithm (path-algorithm)	Fallback path algorithm method (fallback-path-algorithm)	PCE state	Behaviour
				PCE fails (for example, timeout), the secondary path is attempted, skipping the fallback path computation.

In a passive stateful LSP, where the **pce-control** value within the `network-instance default traffic-engineering-policies.policy.segment-list.pce-control` is not configured, the PCC waits for specific events to initiate a return to PCE-based path computation. These events include:

- expiry of the **re-optimization-timer**
- a manual resignal
- expiry of the retry timer, for instance, after a previous path computation failure
- any configuration changes to the segment list (the sequence of segments that constitute the LSP)

When any of these events occurs, the PCC computes a new path using the configured path algorithm (**pce**).

In an active stateful LSP, where the **pce-control** value is within the `network-instance default traffic-engineering-policies.policy.segment-list` , PCC notifies MPLS that the PCE is enabled indicating that the PCC is ready to reuse the PCE for path computations.

11.1.9.11 Configuring fallback path algorithm

Procedure

You can configure the fallback path algorithm by setting the parameter, **fallback-path-algorithm** in the `network-instance.traffic-engineering-policies.policy.segment-list.dynamic` context.

The fallback path algorithm parameter **fallback-path-algorithm** can be configured to **local-cspf** or **none**.

When the fallback path algorithm parameter **fallback-path-algorithm** is set to **none**, ip-to-label path computation is performed, and when set to **local-cspf**, hop-to-label path computation is performed.

Note: The fallback path computation method is applicable only when the path algorithm parameter (`network-instance.traffic-engineering-policies.policy.segment-list.dynamic.path-algorithm`) is set to **pce**.

Example

The following example configures the **fallback-path-algorithm** with the **local-cspf** value:

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering-policies policy srl_te_
policy
    network-instance default {
```

```
        traffic-engineering-policies {
            policy srl_te_policy {
                segment-list 5 {
                    dynamic {
                        path-algorithm pce
                        fallback-path-algorithm local-cspf
                    }
                }
            }
        }
    }
```

11.1.9.12 PCEP Associations

The PCEP Association Groups are used to reference TE-policy uncolored segment list constraints. PCEP Association Groups allow segment lists (LSPs) to share common information, such as common policies or common configuration parameters/constraints. SR Linux routers in the PCC role can reference a set of path constraints while making a path computation request to PCE, instead of passing individual constraints one by one.

The PCEP ASSOCIATION object defines an Association ID and an Association Type to signal any type of association between LSPs (as defined in RFC 8697). Association groups are identified by a tuple consisting of an Association ID, Association Type, and Association Source. Association groups offer a disaggregated approach to specifying association, where the Association ID is equivalent to the path group ID, and the tuple (Association ID, Association Type, Association Source) is equivalent to the path profile ID. The tuple is the key to the association group. Both IPv4 and IPv6 PCEP control planes support signaling of the Association IDs. The PCE uses the Association ID to reference a path profile.

The following are the SR Linux supported PCEP association types:

- Disjoint Association Type, as defined in RFC 8800, is used to signal path diversity constraints
- Policy Association Type, as defined in RFC 9005, is used to signal PCE policies for associating policies

Both of these association types use the *Generic association object with Association ID and Association type* framework (RFC 8697).

11.1.9.12.1 PCE association of segment lists

The PCE associations are applicable to either the `local-cspf` or the `pce` computed segment lists (LSPs), enabling the following behavior:

- Primary, standby, and secondary paths can have different PCE association policies.
- PCE-computed segment lists use PCE association policies as constraints in computation requests and reports.
- In the local-CSPF computed segment list with `pce-control`/`pce-report` configured, the association information is shared with the PCE server in reports.
- In the local-CSPF computed segment list without `pce-control` configured, the PCE-association policies are ignored.
- In IP-to-label and explicit labels, the PCE association policies cannot be configured.

PCEP extensions for LSP association

The PCEP extensions for LSP sets provide a generic mechanism to create a grouping of LSPs in the context of a PCE. This grouping enables the definition of associations between sets of LSPs or between a set of LSPs and a set of attributes.

The PCC can signal one or more ASSOCIATION objects for each Association Type based on the configuration of the PCC-initiated LSP. MPLS passes this information to the PCC module. The PCC can include the ASSOCIATION object and its TLVs in any of the following PCEP messages: PCReq, PCRpt. The PCE may reflect the same ASSOCIATION objects and TLVs in any of the following PCEP messages: PCRep, PCUpd.

The following table displays the base ASSOCIATION object.

Table 12: PCEP ASSOCIATION object and message

Object	Message
ASSOCIATION object	PCRpt, PCReq, PCRep, PCInitiate, PCUpd

The following are the PCEP extensions for IPv4 LSPs:

Table 13: PCEP numbers (IPv4 LSPs)

Value	Name	Object-type		Reference
40	ASSOCIATION	0: Reserved	1: IPv4	RFC 8697

Table 14: PCEP TLV (IPv4 LSPs)

Value	Meaning	Reference
29	Operator-configured Association Range	RFC 8697
30	Global Association Source	
31	Extended Association ID	

The following are the PCEP extensions for IPv6 LSPs:

Table 15: PCEP numbers

Value	Name	Object-type		Reference
40	ASSOCIATION	2: Reserved	2: IPv6	RFC 8697

Table 16: Path Protection Association TLV

Type	Name	Reference
1	Path Protection Association	RFC 8745

Type	Name	Reference
38	Path Protection Association Group TLV	

Table 17: Path Protection Association TLV Flag

Bit	Name	Reference
31	P - PROTECTION-LSP	RFC 8745
30	S - SECONDARY-LSP	
6-29	Unassigned	
0-5	Protection Type Flags	

11.1.9.12.2 Diversity Association Group

The disjoint association signals path diversity constraints using the Disjoint Association Type defined in RFC 8800. The Association ID for the diversity type supports the user-configured mode of operation. The ID value is interpreted as a global value independent of the IP address of the PCC or PCE node that is used to associate segment lists (LSPs).

The DISJOINTNESS-CONFIGURATION TLV is used to describe the diversity parameters requested for the set of segment lists. The PCE uses the DISJOINTNESS-STATUS TLV to convey the status of the diversity parameters after the path computation. The same flags indicate whether any of the parameters were met by the returned path.

Table 18: PCEP TLVs

Type	Name	Reference
46	DISJOINTNESS-CONFIGURATION	RFC 8745
47	DISJOINTNESS-STATUS	

Table 19: Objective functions

Code point	Name	Reference
15	Minimize the number of Shared Links (MSL)	RFC 8800
16	Minimize the number of Shared SRLGs (MSS)	
17	Minimize the number of Shared Nodes (MSN)	

Table 20: NO-PATH-VECTOR bit flags

Code point	Name	Reference
11	Disjoint path not found	RFC 8800
10	Requested disjoint computation not supported	

11.1.9.12.3 Configuring diversity association parameters

You can configure the diversity association parameters in **network-instance* [name] protocols pcep!+ pcc pce-associations diversity*** context.

The following are the parameters for the path Diversity Association:

- **association-id**- a value identifying the association group.
- **association-source** - the source of the association, which can be specified as either an IPv4 or IPv6 address.
- **disjointness-reference** – indicates whether this LSP path is the reference path for the disjoint set of paths. When set, the PCE must first compute the path of this LSP and then apply the requested disjointness type to calculate the path of all other paths in the same diversity association ID.
- **disjointness-type** – specifies the disjointness type, either **strict** or **loose**.
- **diversity-type** – defines the type of diversity, which can be **link**, **node**, **srlg-link**, or **srlg-node**. Configuration of this parameter is mandatory. If this parameter is not configured, the system does not validate the association configuration.

Example

The following example configures the Diversity Association Group parameters:

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols pcep pcc pce-associations diversity
 srl_diversity_association
    network-instance default {
        protocols {
            pcep {
                pcc {
                    pce-associations {
                        diversity srl_diversity_association {
                            association-id 10
                            association-source 10.0.0.3
                            disjointness-reference true
                            disjointness-type strict
                            diversity-type node
                        }
                    }
                }
            }
        }
    }
```

11.1.9.12.4 Policy Association Group

The policy association signals policy constraints using the Policy Association Type defined in RFC 9005.

The Association ID is a globally unique identifier within a specific domain, configured by the operator, and is independent of the source address. Multiple PCC nodes can use the same Association ID in the domain to refer to the same policy or path profile. When a PCC node sends a request to the PCE, the PCE uses the Association ID to look up the corresponding path profile, regardless of the source address of the PCC node. As a result, multiple PCC nodes in the same domain can use the same Association ID to signal different LSPs, and the PCE applies the same path profile to all of them.

Table 21: ASSOCIATION type field

Value	Name	Reference
3	Policy Association	RFC 9005

Table 22: PCEP TLV type indicators

Value	Meaning	Reference
48	POLICY-PARAMETERS-TLV	RFC 9005

11.1.9.12.5 Configuring PCE association policy

Procedure

You can configure the PCE association policy type using the command, **network-instance* [name] protocols pcep! pcc pce-associations policy***. The association ID (`association-id`) is the identifier for the association group and the association source (`association-source`) specifies the source of the association and can be specified as either an IPv4 or IPv6 address.

Example

In the following configuration, the PCC node identifies itself with the IP address 10.0.0.3 when connecting to the PCE and sending requests. The association is uniquely identified by the **association-id** (10), differentiating it from other PCE associations.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols pcep pcc pce-associations policy
 srl_test_association
    network-instance default {
        protocols {
            pcep {
                pcc {
                    pce-associations {
                        policy srl_test_association {
                            association-id 10
                            association-source 10.0.0.3
                        }
                    }
                }
            }
        }
```

```
    }
```

11.1.9.12.6 PCEP ASSOCIATION object error handling

If the ASSOCIATION objects in the PCE do not match the ones sent by PCC in the PCReq or PCRpt messages, the PCC returns an error.

The router handles consistency between the association group configuration for a set of LSPs on a specific PCC. That is, an association group can only have one set of parameters within an association (for example, Diversity Type and Disjointness Type), which ensures that LSPs added to the same association group do not have inconsistent parameters. However, LSPs originating on different PCCs can be added to the same association group, but those association groups have different parameters configured on the different PCCs. In that case, only the NSP can detect parameter inconsistencies.

For LSPs that are delegated and have inconsistent association parameters, the NSP sends a PCUpd down message followed by a PCErr message with the appropriate error message. This causes the affected LSPs to go operationally down. For non-delegated LSPs, the NSP sends a PCErr message.

If an LSP is added to an association group that has inconsistent parameters when compared with the same association group to which operationally up LSPs are already assigned, the NSP only registers an error on the new LSP and leaves the existing LSPs undisturbed.

Table 23: PCEP-ERROR types and names (IPv4)

Error type	Meaning	Error value	Reference
26	Association Error	0: Unassigned	RFC 8697
		1: Association Type is not supported	
		2: Too many LSPs in the association group	
		3: Too many association groups	
		4: Association unknown	
		5: Operator-configured association information mismatch	
		6: Association information mismatch	
		7: Cannot join the association group	
		8: Association ID not in range	

Table 24: PCEP-ERROR types and names (IPv6)

Error type	Meaning	Error value	Reference
26	Association error		
-	-	9: Tunnel ID or endpoints mismatch for Path Protection Association	
		10: Attempt to add another working/ protection LSP for Path Protection Association	
		11: Protection type is not supported	

Table 25: PCEP-ERROR codes (Disjoint association)

Error type	Meaning	Error value	Reference
6	Mandatory Object missing	-	RFC 5440
-	-	15: DISJOINTNESS-CONFIGURATION TLV missing	RFC 8800
10	Reception of an invalid object	-	RFC 5440
-	-	32: Incompatible OF code	RFC 8800

Table 26: PCEP errors (Policy association)

Error type	Meaning	Error value	Reference
26	Association Error	-	RFC 8697
-	-	12: Not expecting policy parameters	RFC 9005
-	Reception of an invalid object	13: Unacceptable policy parameters	RFC 9005

11.1.10 Configuring TE-Policy with admin-group constraints

Procedure

A TE-Policy segment list (LSP) can specify the inclusion or exclusion of a specific admin group for path selection. Admin groups allow operators to define which links should be included or excluded during segment list path computation. For example, avoid including links with long delays or that are heavily used in the path computation of a segment list. Each admin group can support up to 32 values. Admin groups are assigned to TE interfaces and used in the TE-Policy segment list path computation to influence routing decisions.

Example: Associating admin-groups with a TE interface

In the following configuration example, admin groups (`blue` and `red`) are explicitly assigned to a TE interface, enabling targeted control over path computation and routing decisions.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering interface ethernet-1/1.1
    network-instance default {
        traffic-engineering {
            interface ethernet-1/1.1 {
                te-metric 564
                admin-group [
                    blue
                    red
                ]
                interface-ref {
                    interface ethernet-1/1
                    subinterface 1
                }
            }
        }
    }
```

Example: Configuring TE-Policy with admin-group constraints

In the following configuration example, the `exclude-admin-group [ blue ]` statement instructs path computation algorithm (**local-cspf**) to avoid links that have an admin group (`blue`), and the `include-admin-group [ red ]` statement instructs the path computation algorithm to prefer or include links that have an admin group of `red` in the computed path.

```
A:root@srl1# info with-context network-instance default traffic-engineering-policies
 policy srl_uncolored_policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_uncolored_policy {
                policy-type sr-mpls-uncolored
                endpoint 10.1.1.1
                segment-list 3 {
                    segment-list-type primary
                    dynamic {
                        path-algorithm local-cspf
                        te-constraints {
                            exclude-admin-group [
                                blue
                            ]
                            include-admin-group [
                                red
                            ]
```

```
                        }
                    }
                }
            }
        }
    }
```

11.1.11 Uncolored SR-MPLS TE-policy tags

Tags (`tag-set` command) can be configured for an uncolored SR-MPLS TE policy path. A TE policy is a traffic-engineered tunnel that follows a path determined by a sequence of segment routing SIDs. You can customize the binding of traffic to the tunnel based on tags. The tag constraint for a SR-MPLS uncolored TE policy (`sr-mpls-uncolored`) is analogous to the `admin-tag` for SR-TE LSPs in 7x50 SR OS routers.

When a tag-set is assigned to a TE policy, it can contain up to two tags. Services and IP shortcuts use tags to resolve the best transport tunnel. Tunnel resolution configuration can dictate a specific tag or tags; Resolution may be based on the preference of a tag or tags without mandating a tag match. Using tags ensures services and shortcuts resolve to tunnels that match certain criteria, such as hop count, delay, or something similar. Unless a mandatory keyword is used in resolution, preference is given to a more complete match. During resolution, a tag-set with two matching tag-values is preferred over a tag-set with one matching tag-value. Tag match criteria are user configurable and can be set to mandatory, in which case an exact match is required for resolution of network instance or a route to a given TE policy. For example, if a BGP route has a tag-set X, it can only bind to a tunnel with a tag-set X. Otherwise, it remains unresolved.

If no eligible TE-policy exists, the system defaults to regular auto-bind behavior, using LDP, SR-ISIS, or any other lower priority configured tunnel type. Otherwise, the resolution fails.

If other transport tunnel options, like LDP or SR-ISIS, are available, they are used instead. If no alternative transport method is available,the service or route would remain inactive.

Tag-sets enable the system to resolve to specific transport tunnels (or groups of eligible transport tunnels) for BGP routes for applications such as BGP labeled unicast, IP-VRF, or EVPN. Additionally, tag-sets specify a finer level of granularity on the next-hop or the far-end prefix associated with a BGP labeled unicast route or unlabeled BGP route shortcut tunnels.

11.1.11.1 Associating an SR-MPLS TE policy with a tag-set

Procedure

An uncolored SR-MPLS TE policy can be configured with tag-sets. The tag-set must have at least one tag value and can have a maximum of two values configured.

Example: Configure a tag-set

In the following example, the tag-set (`srl-tag-set`) is configured with tag values of 2 and 4.

```
--{ * candidate shared default }--[ ]--
# info with-context routing-policy tag-set srl-tag-set
    routing-policy {
        tag-set srl-tag-set {
            tag-value [
                2
                4
```

```
            ]
        }
    }
```

Example: Associate an SR-MPLS TE policy with a tag-set

In the following example, the uncolored SR-MPLS TE policy (`sr-mpls-te-policy`) is associated with a tag-set (`srl-tag-set`).

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies policy sr-mpls-
te-policy
    network-instance default {
        traffic-engineering-policies {
            policy sr-mpls-te-policy {
                tag-set srl-tag-set
            }
        }
    }
```

11.1.11.2 Associating a tag-set with BGP next-hop resolution

Procedure

By associating a tag-set, you can constrain the tunnels used by the system for resolution of BGP next-hops or prefixes and BGP labeled unicast routes.

Example

The following configuration example sets a tag set for BGP next hop resolution.

```
--{ * candidate shared default }--[ ]--
# info with-context routing-policy policy srl-test-policy default-action bgp
    routing-policy {
        policy srl-test-policy {
            default-action {
                bgp {
                    next-hop-resolution {
                        set-tag-set srl-tag-set
                    }
                }
            }
        }
    }
```

11.1.11.3 Configuring a tag as mandatory for BGP route resolution

Procedure

Uncolored SR-MPLS TE-policy tag match criteria are user configurable and can be set to mandatory, in which case an exact match is required for the resolution of a network instance or a route to a TE Policy. Routing policy configures tag-set and match is via uncolored SR-MPLS TE-policy tag. Configuring SR-MPLS TE-policy tags allows for a more detailed level of granularity when determining the next hop associated with a BGP-labelled unicast route, ensuring that only routes meeting specific policy criteria are selected.

Example

The following configuration shows how to make the uncolored SR-MPLS TE-policy tag mandatory for BGP route resolution. The next-hop resolution uses IPv4 next-hops, and the allowed tunnel type is an SR-MPLS uncolored TE-policy.

```
--{ * candidate shared default }--[ ]--
# info with-context network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                admin-state enable
                autonomous-system 78
                afi-safi l3vpn-ipv4-unicast {
                    ipv4-labeled-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                tunnel-resolution {
                                    allowed-tunnel-types [
                                        te-policy-sr-mpls-uncolored
                                    ]
                                    selection-attributes {
                                        tag {
                                            mandatory true
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

11.1.12 Uncolored SR-MPLS TE path failure codes

The table below lists the uncolored SR-MPLS TE path failure codes and their meanings.

Table 27: Uncolored SR-MPLS TE path failure codes

Failure codes	Description
path-computation-request-timeout	Path computation request timed out
path-computation-no-route	No valid route is returned for path computation request
no-resources-available	Required resources are depleted, not enough resources to establish the requested segment-list
path-computation-bad-node	Path computation failure due to a resolution issue of one or more of the hops
path-computation-routing-loop	Path computation failure due to routing loop
unknown	Segment-list is down because of unknown reason

Failure codes	Description
path-computation-no-route-owner	Path computation failure as none of the IGP instances had a valid route to one of the hops
path-computation-hop-limit-exceeded	Path computation failure due to hop limit. No path within the hop limit constraint configured
srlg-not-disjoint	SRLG is shared with primary segment-list and there is no other viable path with dispersed SRLG
srlg-not-computed-path	SRLG is not applicable, as primary segment-list has no applicable SRLG for path computation
srlg-primary-segment-list-down	SRLG is not applicable, as primary segment-list is down
unresolved-first-segment	The router is unable to resolve the first SID (MPLS label value) into one or more outgoing interfaces and next-hops
fib-add-pending	Segment-list is kept down, when adding next-hop into the FIB
fib-add-failed	FIB has failed to add the next-hop group. Next-hop group represents a group of next-hops for valid segment-lists under a TE-policy
maximum-label-stack-depth-exceeded	The resolution of the named path requires more labels than supported by the datapath
pce-update-with-empty-ero	PCE update has empty Explicit Route Object (EROs)
segment-list-admin-down	Segment-list is administratively down
ipv4-hops-in-ipv6-path	IPv4 and IPv6 hops are mixed in explicit path
ipv6-hops-in-ipv4-path	IPv6 and IPv4 hops are mixed in explicit path
sid-hops-in-ip-path	SID (label-based) and IP hops are mixed in explicit path
sid-hops-with-invalid-path-computation	SID hops (labeled hops) with path computation local-cspf/pcep is not allowed
policy-down	Traffic engineering policy is down
pce-association-conflict	PCE-association conflict
retry-on-config-change	Segment-list retry attempted due to config change
clear-command	Segment-list retry attempted due to manual clear command
secondary-segment-list	Secondary type segment-list, primary is always preferred when available
bfd-down	BFD is reported down

Failure codes	Description
te-rtr-id-not-configured	TE router ID config is missing
pce-down	PCE is unavailable
pce-error	PCE response has error or timed-out
pcc-error	PCC responded with error
delay-metric-limit-exceeded	Segment-list delay metric limit exceeded

11.1.13 LSP statistics

SR Linux supports the collection of segment list (LSP) statistics for uncolored TE Policies. When migrating to Software-Defined Networking (SDN) infrastructure, the LSP statistics provide essential data that enable an external Path Computation Element (PCE) to perform path computation. LSP statistics offer real-time utilization insights, which can be used while re-optimizing paths in response to congestion.

To maximize scalability, statistics are collected per TE Policy and identified by a stats index. The statistics collected do not provide a breakdown by forwarding class. In aggregate mode, the statistics collection is supported in both ingress and egress directions for the active segment list within the TE-Policy. One aggregate counter per uncolored TE Policy is supported.

Statistics are maintained per policy and include ingress and egress configurations. Both ingress and egress statistics are displayed at the policy level.

For an active segment list, the statistics collection behavior is as follows: if a non-primary segment list becomes active, counting continues with the new active segment list. The same stats index is assigned to all segment lists (LSPs) that constitute the TE Policy.

If the allocated statistics resources are exhausted and additional TE policies require statistics collection, the affected TE Policy candidate path or segment list STATE reports a statistics collection failure. When resources are exhausted, TE policies that require statistics collection are placed on a retry list. As statistics resources become available, the system retries statistics collection for these entries.

The allocation of statistics resources is based on the statistics configuration. The system reserves statistics resources (for example, ingress or egress statistic IDs) upfront based on the configuration.

The operational state of the BSID or tunnel does not change or reset the statistics. If the policy is administratively disabled or the statistics configuration is disabled, statistics resource is released, and counters are reset. The statistics collection continues without interruption for any binding-sid change or active segment list change.

On 7250 IXR Gen 2c+ devices, LSP statistics require bank allocation in datapath to host the collected statistics. For more information about counter bank allocation, see the *Flexible counter bank allocation* section in the *SR Linux Configuration Basics Guide*.

Supported platforms

LSP statistics is supported on the following platforms:

- 7730 SXR
- 7250 IXR Gen 2c+
- 7250 IXR Gen 3

11.1.13.1 Configuring LSP statistics

Procedure

LSP statistics provide real-time utilization data, essential for optimizing path computation in Software-Defined Networking (SDN) infrastructure that depend on an external PCE for path computation. To configure the LSP statistics, use the **network-instance.traffic-engineering-policies.policy.statistics** command.

Example: Configuring LSP statistics for uncolored TE-Policy

The following configuration example enables both ingress and egress statistics collection for `srl_test_policy` uncolored TE Policy.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default traffic-engineering-policies policy srl_test_
policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_test_policy {
                policy-type sr-mpls-uncolored
                admin-state enable
                endpoint 10.0.0.3
                statistics {
                    ingress {
                        admin-state enable
                    }
                    egress {
                        admin-state enable
                    }
                }
                segment-list 1 {
                    admin-state enable
                    segment-list-type primary
                    dynamic {
                        path-algorithm local-cspf
                    }
                }
            }
        }
    }
```

11.2 Colored TE policy

RFC 9256 describes the concept of a colored TE policy. This policy describes a source-routed path from a head-end router to an endpoint, and it determines which traffic flows should be steered through that path. A colored TE-Policy for a specific head-end router can be statically configured on that router or advertised to it as a BGP route.

A colored TE policy is identified by the tuple of <head-end, color, endpoint>. Each colored TE policy is associated with a set of one or more candidate paths, one of which is selected to implement the colored TE policy and is installed in the data plane. The colored TE policies have additional attributes, such as preferences, binding SIDs, and segment lists, which are used in the policy selection and implementation.

SR Linux colored TE policy supports 32 programmed segment lists per candidate path. Each segment list can be assigned a weight to influence its share of traffic compared to other segment lists of the same policy.

For weighted ECMP across segment lists within a given candidate path, the weight can be configured at the segment list level. Segment lists that traverse higher bandwidth links can be assigned greater weights, allowing the hash algorithm to map more flows to those segment lists.

Supported platforms

Colored TE policy is supported on the following platforms:

- 7250 IXR Gen 2
- 7250 IXR Gen 2c+
- 7730 SXR
- 7250 IXR Gen 3

11.2.1 Basic concepts

Head-end

The head-end is an IP address that identifies the router at the source of the source-routed path.

Endpoint

The endpoint is an IP address that identifies the router that is the destination of the source-routed path.

Color

Color is a label assigned to a set of traffic flows or routes. Color is a property that determines the sets of traffic flows that the policy should steer. It is a mandatory parameter for a colored TE policy.

Discriminator

A discriminator is a unique value assigned to each policy to differentiate between multiple policies with the same color. It is a mandatory parameter for both local and non-local policies.

Candidate paths

A colored TE policy can define one or multiple candidate paths. Each candidate path comprises a set of one or more segment lists that load-balance the traffic toward the endpoint. The traffic distribution is based on each segment list's relative weights. Each candidate path has a preference value. For a specific (head-end, endpoint, and color) tuple, the candidate path with the highest preference is selected as the active path and installed into the datapath. A colored TE policy can contain multiple candidate paths, but only one is chosen as the best and active path. The control plane retains the lower preference candidate paths so the next best one can take over if the active path becomes unavailable.

Binding SID (BSID)

Each candidate path can have a binding SID (BSID), as per RFC 9256. The SR Linux implementation requires every signaled and configured candidate path to have a BSID. The BSID opaquely represents the entire candidate path to upstream routers.

The binding SID must be an available label (MPLS) in the reserved label block associated with colored TE policies; otherwise, the policy cannot be activated.

Typically, all candidate paths within a colored SR TE-Policy should be assigned the same BSID as described in RFC 9256, but this is not enforced or guaranteed.

11.2.2 Best path selection process

Procedure

Step 1. Candidate paths under a TE policy are validated and ranked based on the following criteria, in strict order of priority:

- **a.** Candidate path preference: Higher values are preferred.
- **b.** Protocol origin: Higher values are preferred.
- **c.** Originator (ASN, Node-address): Lower values are preferred.
- **d.** Discriminator: Higher values are preferred.

Step 2. After sorting the candidate paths based on the defined priority criteria, each path is evaluated sequentially. The first path with a successful BSID allocation is selected as the best path.

Step 3. The selected path from step 2 is added to the tunnel table.

11.2.3 Statically-configured colored TE policies

Colored TE policies can be statically configured locally on a head-end node or dynamically learned by a head-end node through the BGP colored TE policy route.

When the **head-end** parameter in `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] color?` is set to **local**, a static policy is local, and when it is set to an IP address, it is non-local.

11.2.3.1 Configuring a static local colored TE policy

Procedure

To statically configure a local colored TE policy, set the parameter **head-end** to **local** within the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] color?` context.
To configure a local colored TE policy, perform the following tasks:

1. Configure a static label range.
2. Assign the static MPLS label range to the BSID.
3. Configure the local colored TE policy using the defined BSID.

Example: Configuring a static label range.

Use the command, `system.mpls.label-ranges` to configure the static label range. All participating routers within the SR domain use the static label range. In the configuration example, a static MPLS range (`srl-static-label-range`) is set from 16001-104857.

```
--{ + running }--[  ]--
# info with-context system mpls label-ranges static srl-static-label-range
    system {
        mpls {
            label-ranges {
                static srl-static-label-range {
                    shared false
                    start-label 16001
                    end-label 104857
                }
            }
        }
    }
```

Example: Assigning a static MPLS range to the BSID

The following example configures the BSID in SR-capable routers to align with the configured label range (`srl-static-label-range`).

```
--{ + running }--[  ]--
# info with-context network-instance default traffic-engineering-policies binding-sid
    network-instance default {
        traffic-engineering-policies {
            binding-sid {
                static-label-block srl-static-label-range
            }
        }
    }
```

Example: Configuring a static local colored TE policy

The following example configures a static local colored TE policy (`srl_local_colored_policy`) using the defined BSID. The MPLS label value 16001 falls within the defined MPLS label range (`srl-static-label-range`) and serves as the BSID for the static local colored TE policy.

The parameter **head-end** is set to **local** value.

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies policy srl_
local_colored_policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_local_colored_policy {
                policy-type sr-mpls-colored
                color 600
                endpoint 192.0.2.5
                discriminator 78
                head-end local
                binding-sid {
                    mpls-label 16001
                }
            }
        }
    }
```

11.2.3.2 Configuring a static non-local colored TE policy

Procedure

To statically configure a non-local colored TE policy, set the parameter **head-end** to an IPv4 or IPv6 address within the `network-instance* [name] traffic-engineering-policies!+ policy* [policy-name] color?` context.

Example

The following example configures a static non-local colored TE policy (`srl_non_local_colored_policy`). The parameter **head-end** is set to an IPv4 address (`10.10.10.1`).

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default traffic-engineering-policies policy srl_non_
local_colored_policy
    network-instance default {
        traffic-engineering-policies {
            policy srl_non_local_colored_policy {
                policy-type sr-mpls-colored
                color 600
                endpoint 192.0.2.5
                discriminator 78
                head-end 10.10.10.1
                binding-sid {
                    mpls-label 16001
                }
            }
        }
    }
```

11.2.4 BGP signaled colored TE policy

MP-BGP (Multi-Protocol BGP) is used to advertise colored TE policies between routers. A controller or another BGP speaker can instruct a BGP edge router to steer traffic according to a specific colored TE policy; it can advertise the colored TE policy candidate path through MP-BGP.

The Network Layer Reachability Information (NLRI) within the UPDATE messages of the AFI/SAFIs identifies a colored TE policy candidate path. The NLRI format of this MP-BGP route is described in *draft-ietf-idr-sr-policy-safi-13*.

The colored TE policy information is carried using the AFI/SAFI combination of AFI=1, SAFI=73 for IPv4 destinations and AFI=2, SAFI=73 for IPv6 destinations.

SR Linux AFI-SAFI name for IPv4 is `srte-policy-ipv4`, and the AFI-SAFI name for IPv6 is `srte-policy-ipv6`.

To exchange routes that belong to the (AFI=1, SAFI=73) or (AFI=2, SAFI=73) address family with a specific BGP neighbor, the family configuration for that neighbor must include the `srte-policy-ipv4` or `srte-policy-ipv6` AFI-SAFI names.

When BGP router receives a `srte-policy-ipv4` route (AFI=1, SAFI=73) or a `srte-policy-ipv6` (AFI=2, SAFI=73) from a peer, it runs its standard BGP best path selection algorithm to choose the best path for each NLRI combination of discriminator, endpoint, and color. If the best path is targeted to this router as the head-end, BGP extracts the colored TE policy details into the local database.

Multiple candidate paths can exist for a colored TE policy, although only one path can be selected as the best path of the colored TE policy and become the active path. If several candidate paths of the same colored TE policy (endpoint, color) are advertised via BGP colored TE policy to the same head-end, unique discriminators for each NLRI are recommended. For information about BGP colored TE policy traffic steering, see Colored TE policy traffic steering.

Note:
A BGP colored TE policy route is considered malformed if it does not have at least one segment list TLV, which triggers error-handling procedures such as session reset or treat-as-withdraw.

11.2.4.1 Importing a static non-local colored TE policy

Procedure

SR Linux supports the origination of non-local colored TE policy routes configured as static candidate paths. Static non-local policies are signaled as BGP NLRIs.

To add non-local IPv4-endpoint static candidate paths that are maintained by the SR policy manager to the BGP RIB-IN for AFI=1/SAFI=73 routes, set the parameter, `import-static` to `true` using the `network-instance.protocols.bgp.afi-safi.srte-policy-ipv4` command. Similarly, to add non-local IPv6-endpoint static candidate paths that are maintained by the SR policy manager to the BGP RIB-IN for AFI=2/SAFI=73 routes, set the parameter, `import-static` to `true` using the `network-instance.protocols.bgp.afi-safi.srte-policy-ipv6` command.

Note:

- If the head-end address is IPv4, it is encoded inside an IPv4-address-specific route-target extended community (type/subtype 0x4102).
- If the head-end address is IPv6, it is encoded inside an IPv6-address-specific route-target extended community (path attribute 25, extended community type/subtype 0x0002).

Example: Importing non-local colored TE policy (IPv4)

The following example imports a non-local IPv4-endpoint static candidate path.

```
--{ + candidate shared default }--[ ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast srte-
policy-ipv4 import-static
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    srte-policy-ipv4 {
                        import-static true
                    }
                }
            }
        }
    }
```

Example: Importing non-local colored TE policy (IPv6)

The following example imports a non-local IPv6-endpoint static candidate path.

```
--{ + candidate shared default }--[ ]--
```

```
# info with-context network-instance default protocols bgp afi-safi ipv6-unicast srte-
policy-ipv6 import-static
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv6-unicast {
                    srte-policy-ipv6 {
                        import-static true
                    }
                }
            }
        }
    }
```

11.2.4.2 Colored TE policy encoding using TLV

When a BGP router advertises a colored TE policy candidate path, the specific details are encoded within the **Tunnel Encapsulation Attributes** as defined in RFC 9012, using a Tunnel-Type called SR Policy Type with codepoint 15.

The following sub-TLVs are defined for use with TLV 15:

- Preference subTLV (type 12) - optional, encodes 4-byte value
- Binding SID subTLV type (type 13) - optional, encodes MPLS label value
- Segment-List subTLV (type 128) - optional, can be repeated multiple times, encodes a segment list as collection of subsubTLVs:
  - Weight subsubTLV (type 9) - optional, encodes 4-byte value; zero is invalid
  - Segment subsubTLV (type dependent on Segment Type) - optional, encodes a single segment in a segment-list
- Explicit Null Label Policy subTLV (type 14) - currently not supported in SR Linux
- Policy Priority subTLV (type 15) - currently not supported in SR Linux
- Policy Candidate Path Name subTLV (type 129)
- Policy Name subTLV (type 130) - currently not supported in SR Linux

Note: A BGP colored TE policy route is considered malformed and triggers an error-handling procedure such as treat-as-withdraw if it uses any other TLV or multiple TLVs with code 15.

11.2.4.3 Colored TE policy traffic steering

About this task

When a BGP router that supports the new AFI-SAFI receives a BGP message encoding a colored TE policy candidate path, it follows these steps to steer traffic:

Procedure

Step 1. The router applies the BGP import policy to process the received candidate path.

Step 2. The router runs the BGP best path selection algorithm to choose one best BGP path for each NLRI combination of <discriminator, color, endpoint>. See Validating colored TE policy routes.

Step 3. If the selected best path for a particular <discriminator, color, endpoint> tuple is targeted to this router as head-end, BGP extracts the candidate path details and forwards them to an SR policy manager. The SR policy manager maintains a lists all candidate paths from various sources, including static configuration, and BGP.

Step 4. The SR policy manager evaluates the validity of each candidate path and updates its status as conditions change. For information on validation process, see Validating colored TE policy routes.

Step 5. For each <color, endpoint>, the SR policy manager selects the highest-preference valid candidate path for installation into the data path. If multiple valid candidate paths have the same best preference, tie-breaking rules determine the selection.

Step 6. The SR policy manager programs the selected candidate path from the previous step into the datapath, resulting in the creation of the following entries:

a. An Incoming Label Map (ILM) entry that matches the BSID and binds it to a Next Hop Group (NHG) containing one Next-Hop Label Forwarding Entry (NHLFE) for each of the valid segment lists.

b. A colored tunnel entry that binds to an NHG containing one NHLFE for each of the valid segment-lists. The color of the tunnel matches the <color> of the active candidate path from which it is derived.

11.2.4.3.1 Validating colored TE policy routes

BGP selects the best path for every TE policy NLRI, keyed by the combination of <discriminator, color, endpoint>. The best path selection algorithm uses the normal BGP rules. BGP excludes statically configured TE policies when selecting the best path.

Note:
None of the Tunnel Encapsulation attribute values are used for best path selection.

If the best path is targeted to this router as the head-end, BGP extracts the colored TE policy details into the SR policy manager. A BGP colored TE- Policy route is deemed to be targeted to this router as the head-end if either:

- it has no route-target extended community and a NO-ADVERTISE standard community
- it has an IPv4 address-specific route-target extended community with an IPv4 address matching the system IPv4 address of this router

Validation process

After the candidate path is passed to the SR policy manager, it validates the candidate path. The SR policy manager determines whether the candidate path is valid or invalid and continuously updates as conditions change.

A segment list is valid if all of the following conditions are met:

- It is not empty (0 segments).
- BSID is present and is within the expected range of MPLS labels.
- If weight is specified, it is non-zero (zero-weight paths are not valid).
- It consists only of type A segments, and the number of segments does not exceed datapath capabilities.
- The SR policy manager performs path resolution for the first SID in one or more outgoing interfaces and next-hops.

- When selecting the candidate path for each <color, endpoint>, the following criteria are applied in order of priority:
  1. Highest preference: If no preference sub-TLV is present in the BGP route, a default preference value of 100 is used.
  2. Highest protocol origin: The order of preference is:
     a. Static (30)
     b. BGP (20)
- Lowest originator value: This is a 160-bit value created by combining the origin AS and the 128-bit node address. The origin AS is the first AS in the AS_PATH of the route (or the peer AS if the AS_PATH is empty), and the node address is the IPv4 or IPv6 address of the originator, extracted from BGP Originator ID (RFC4456) or Router ID of the peer (from its OPEN message).
- Highest discriminator value

11.2.4.4 Colored TE policy data forwarding

The following scenarios describe how a BGP router forwards traffic according to a colored TE policy candidate path that has been installed in its datapath:

- An MPLS packet is received with a top label that matches an ILM entry programmed in step 6a. In this case, the binding SID label is popped, and the next forwarding step operates on an MPLS packet with N new labels pushed on top of the stack; the N labels correspond to the NHLFE/segment list selected by the ECMP load-balancing hash algorithm. The packet is forwarded out of the box according to the ILM entry for the top label of the new label stack. Depending on the SID instruction, this ILM entry can pop or swap the top label.
- An IP packet is received and matches a BGP route with one or more of its next-hops resolved by a colored tunnel entry as programmed in step 6b. Assuming that the ECMP load-balancing hash algorithm selects one of these next hops, the forwarding pipeline is given an MPLS packet with N labels corresponding to the selected NHLFE/segment list. The packet is forwarded out of the box according to the ILM entry for the top label of the new label stack. Depending on the SID instruction, this ILM entry can pop or swap the top label.
- An IP packet or Ethernet frame is received in the context of an IP-VRF or MAC-VRF and matches a VPN route that has one or more of its next-hops resolved by a colored tunnel entry as programmed in step 6b. Assuming that the ECMP load-balancing hash algorithm selects one of these next-hops, the forwarding pipeline is given an MPLS packet with N labels corresponding to the selected NHLFE/segment list. The packet is forwarded out of the box according to the ILM entry for the top label of the new label stack. Depending on the SID instruction, this ILM entry can pop or swap the top label.

11.2.4.5 Next-hop address encoding for colored TE policy

When BGP re-advertises a colored TE policy route, the BGP next-hop encoding depends on the peer-type and configuration, as follows:

- When `next-hop-self` is enabled for a BGP session with peer-X, the BGP next-hop address in the RIB-IN is overwritten with the local IP address associated with the session. The resulting BGP next-hop address can be either an IPv4 or IPv6 address, depending on the type of transport peer used for the session.

- When `next-hop-self` is not enabled for a session with an IBGP peer (peer-X), the next-hop address is not modified and remains the same as the original next-hop address.
- When `next-hop-self` is not enabled for a session with an EBGP peer (peer-X), the next-hop address is not modified. It remains the same as the original next-hop address, which has been overwritten with the local IP address associated with the session. The resulting BGP next-hop address can be either an IPv4 or IPv6 address, depending on the type of transport peer used for the session.

11.2.5 LSP statistics

SR Linux supports the collection of segment list (LSP) statistics for colored TE Policies. The statistics collection is supported in both ingress and egress directions for the active candidate path within the colored TE-Policy. A stats index identifies the statistics collected and one stats resource is allocated to all segment lists under the candidate path. Statistics collection is enabled directly under the colored TE-Policy, and stats collection follows the best path selection algorithm. Statistics collection is through the active candidate path and all its active segment lists.

The stats index is not persistent. When a candidate path becomes operationally down, such as when the number of active segment lists falls below a configured threshold and then later returns to the operationally up state, the associated statistics counters for that candidate path are reset to zero.

On 7250 IXR Gen 2c+ devices, LSP statistics require bank allocation in datapath to host the collected statistics. For more information about counter bank allocation, see the *Flexible counter bank allocation* section in the *SR Linux Configuration Basics Guide*.

Supported platforms

LSP statistics is supported on the following platforms:

- 7730 SXR
- 7250 IXR Gen 2c+
- 7250 IXR Gen 3

Ingress statistics

You can configure ingress statistics at the TE-Policy level. The statistics counters are aggregated and populated into the YANG state at the policy level, and not separately for each candidate path.

Ingress stats index allocation is based on the best path selection algorithm. If the best path selected has the ingress statistics enabled, then the corresponding stats index is reserved and used in an Incoming Label Map (ILM) entry. If no best path is selected, the stats index resource is released.

Note:
If we have two candidate paths, CP1 and CP2, with the same BSID, and CP1 has statistics enabled, while CP2 does not. If CP1 is selected as the best path, the corresponding ILM entry includes statistics. If CP2 is chosen as the best path, the ILM entry will not have statistics associated with it.

Egress statistics

You can configure egress statistics at TE-Policy level. The statistics counters are aggregated and populated into the YANG state at the segment list level.

When a candidate path with egress statistics enabled is selected for set up, all valid segment lists under the candidate path allocate their own egress stats index.

11.2.5.1 Configuring LSP statistics

Procedure

To configure the LSP statistics, use the **network-instance.traffic-engineering-policies.policy.statistics** command.

Example: Configuring LSP statistics for colored TE-Policy

The following configuration example enables both ingress and egress statistics collection for `srl_local_colored_policy` colored TE-Policy.

```
--{ + running }--[  ]--
# info with-context network-instance default traffic-engineering-policies policy srl_
local_colored_policy statistics
   network-instance default {
        traffic-engineering-policies {
            policy srl_local_colored_policy {
                policy-type sr-mpls-colored
                statistics {
                    ingress {
                        admin-state enable
                    }
                    egress {
                        admin-state enable
                    }
                }
            }
        }
    }
```

11.3 Non-Stop Routing (NSR) for TE-Policy (uncolored or colored)

TE-Policy NSR is a high-availability feature that enables a router to continue forwarding traffic without disruption during a control plane switchover, such as when the primary routing processor (CPM) fails or is manually restarted.

With NSR enabled on the SR Linux routers, TE-Policy segment lists and candidate paths in case of colored TE-Policy remain intact in case of any faults in the *Traffic Engineering Manager* process. If a fault occurs, a reliable and deterministic switchover to the inactive control complex occurs, ensuring that the traffic engineering database and TE-Policy remain unaffected.

NSR achieves high availability through parallelization by always maintaining up-to-date routing state information about standby `te_policy.mgr` instances. TE-Policy (uncolored or colored) NSR is always enabled by default and does not require configuration.

For more information about NSR, see the “Non-Stop Routing” section of the SR Linux Configuration Basics Guide.

Supported platforms

Non-Stop Routing (NSR) for TE-Policy (uncolored or colored) is supported on the following platforms:

- 7250 IXR-6e
- 7250 IXR-10e
- 7250 IXR Gen 3

12 Flexible algorithm (Flex-Algo) for IS-IS

Segment routing with flexible algorithms (Flex-Algo) is an advanced networking technique that enhances the efficiency and performance of network routing. Flex-Algo enables the definition of custom routing policies based on multiple constraints such as latency, bandwidth, reliability, and administrative requirements beyond just the shortest path. This adaptability allows networks to handle diverse applications with varying requirements, such as real-time video, large data transfers, and mission-critical communications. Flex-Algo also enables more efficient network utilization, improved quality of service, and the ability to meet specific service level agreements.

Flex-Algo — as described in RFC 9350 — provides a way for IGPs to compute constraint-based paths across a domain. Flex-Algo uses extensions to IS-IS to advertise TLVs containing one or more flexible algorithm definitions (FADs). Each FAD is associated with a numeric identifier and identifies a set of metrics and constraints to calculate the best path along the constrained topology.

Flex-Algo works by incorporating additional attributes into the routing decision process. For example, a network path might be chosen based on its low latency for a video conferencing application, or on its high bandwidth for a large file transfer. This capability allows networks to dynamically adapt to changing conditions and demands.

When used with segment routing, one or more prefix node-SIDs can be associated with a Flex-Algo identifier, thereby providing a level of traffic engineering without any associated control plane overhead or additional label stack imposition. The classic SPF technology used for shortest path calculation is referred to as algorithm 0.

SR Linux supports Flex-Algo with IS-IS and segment routing, specifically, SR-MPLS and SRv6.

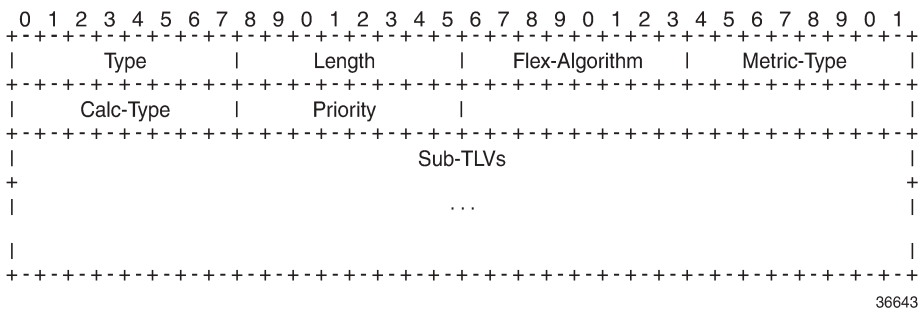
12.1 Flexible algorithm definition (FAD)

A FAD is the construct that identifies how a path for a flexible algorithm will be computed, and consists of three components:

- a calculation type
- a metric type
- a set of constraints, such as include or exclude statements

To guarantee loop-free forwarding for paths computed with Flex-Algo, all routers that participate in a flexible algorithm must receive its definition. In IS-IS, the definition of the flexible algorithm is advertised using the FAD sub-TLV, which is a sub-TLV of the Router Capability TLV and has area scope, as shown in the following diagram.

Figure 8: IS-IS FAD sub-TLV



The Flex-Algo field contains a numeric identifier in the range 128 to 255 that is associated with the FAD through configuration. The Metric-Type field contains one of IGP metric (0), Min Unidirectional Link Delay (1), or TE Default Metric (2). The Calc-Type field contains a value from 0 to 127, identifying the IGP algorithm type, such as shortest path (0). One or more sub-TLV fields may be present to specify colors that are used to include or exclude links during the Flex-Algo path computation. These are encoded using Include-any Admin Group, Include-all Admin-Group, and Exclude Admin Group sub-TLVs.

The Sub-TLV field may also contain a Flags sub-TLV. In *RFC 9350*, only the M-flag (Prefix Metric) is defined. The M-flag indicates that the Flex-Algo Prefix Metric (FAPM) sub-TLV must be advertised with the prefix. The FAPM is not a sub-TLV of the FAD, but rather a sub-TLV of the Extended IP Reachability TLV, and is intended to assist with inter-area and inter-domain Flex-Algo path calculations.

Any IGP shortest-path tree calculation is limited to a single area, and the same applies to Flex-Algo. To allow for inter-area or inter-domain Flex-Algo calculations, the FAPM sub-TLV can be attached to Extended IP Reachability TLVs that are advertised between areas or domains. The FAPM sub-TLV contains the metric equivalent to the metric of the redistributing router to reach the prefix. If the FAD Flags sub-TLV has the M-flag set, the FAPM must be used when calculating prefix reachability for inter-area and inter-domain prefixes.

Only a subset of the routers participating in each flexible algorithm need to advertise the definition of the flexible algorithm. However, every router that is part of the intended Flex-Algo topology must be configured to participate in the flexible algorithm and all participating routers must use the same identical flexible algorithm. If a router is not configured to participate in a specific flexible algorithm, it ignores FAD sub-TLV advertisements for that flexible algorithm.

12.2 Application-specific link attributes

Advertisement of link attributes for the purpose of traffic engineering was initially introduced by RFC 5305, which included a number of sub-TLVs encoded within the Extended IS Reachability TLV, such as admin group, TE default metric, maximum link bandwidth, and unreserved bandwidth.

RFC 7308 updated RFC 5305 by increasing the size of the admin group sub-TLV, thereby allowing for advertisement of more than the standard 32 admin groups per link. RFC 5305 was again updated by RFC 8570, which proposed the use of metric extensions, adding additional sub-TLVs to the Enhanced IS Reachability TLV, such as unidirectional link delay, unidirectional link loss, and unidirectional available bandwidth. These traffic-engineering extensions have been widely deployed for RSVP-TE purposes.

Other applications that also make use of traffic-engineering link attributes have been defined, such as SR, Loopfree Alternates (LFAs), and Flex-Algo. If these applications coexist, it may be advisable to

unambiguously indicate which traffic-engineering attributes apply to which application. Their requirements may differ on a link-to-link basis, or two applications may not be fully congruent; for example, SR may not be fully deployed network-wide. For these reasons, Flex-Algo specifies the use of Application-Specific Link Attributes (ASLAs), from RFC 8919, which defines two code points for IS-IS ASLA advertisements:

- ASLA sub-TLV for Extended IS Reachability and Neighbor Link Attributes TLVs (TLVs 22, 23, 25, 141, 222, and 223)
- Application-Specific Shared Risk Link Group (SRLG) TLV (not supported on SR Linux)

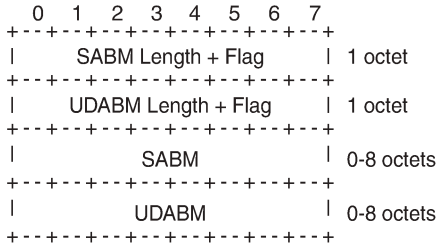
The ASLA sub-TLV contains Link Attribute sub-sub-TLVs, the format of which matches the existing formats defined in RFC 5305, RFC 7308, and RFC 8570.

The ASLA sub-TLV advertisements are coupled with an Application Identifier Bit Mask that identifies the applications associated with an advertisement. Two bit masks are available for use:

- The Standard Applications Bit Mask (SABM) is used for applications, where the definition of each bit is controlled by IANA (for applications such as flexible algorithm).
- The User-Defined Applications Bit Mask (UDABM) allows for future non-standard extensibility.

The encoding shown in the following diagram is used by both the ASLA sub-TLV and the Application-Specific SRLG TLV.

Figure 9: Application Identifier Bit Mask



The SABM Length + Flag field contains a single L-flag, known as the Legacy flag. When the L-flag is set in the Application Identifier Bit Mask, all the applications specified in the bit mask must use the legacy traffic-engineering advertisements for the corresponding link. That is, link attributes should be carried as sub-TLVs of the Extended IS Reachability TLV rather than sub-sub-TLVs of the ASLA sub-TLV. This allows for a level of backward compatibility such that legacy advertisements may continue to be used if:

- Only RSVP-TE is deployed.
- Only SR/LFA is deployed.
- A combination of RSVP-TE and SR/LFA is deployed, but the set of links that each application uses are fully congruent.

The UDABM Length + Flag field contains a single R-flag, which is reserved for future use.

The SABM field defines four bits to identify applications:

- R-bit (bit 0) specifies RSVP-TE
- S-bit (bit 1) specifies SR
- F-Bit (bit 2) specifies LFA
- X-bit (bit 3) specifies Flex-Algo

12.3 Applicability of Flex-Algo to segment routing

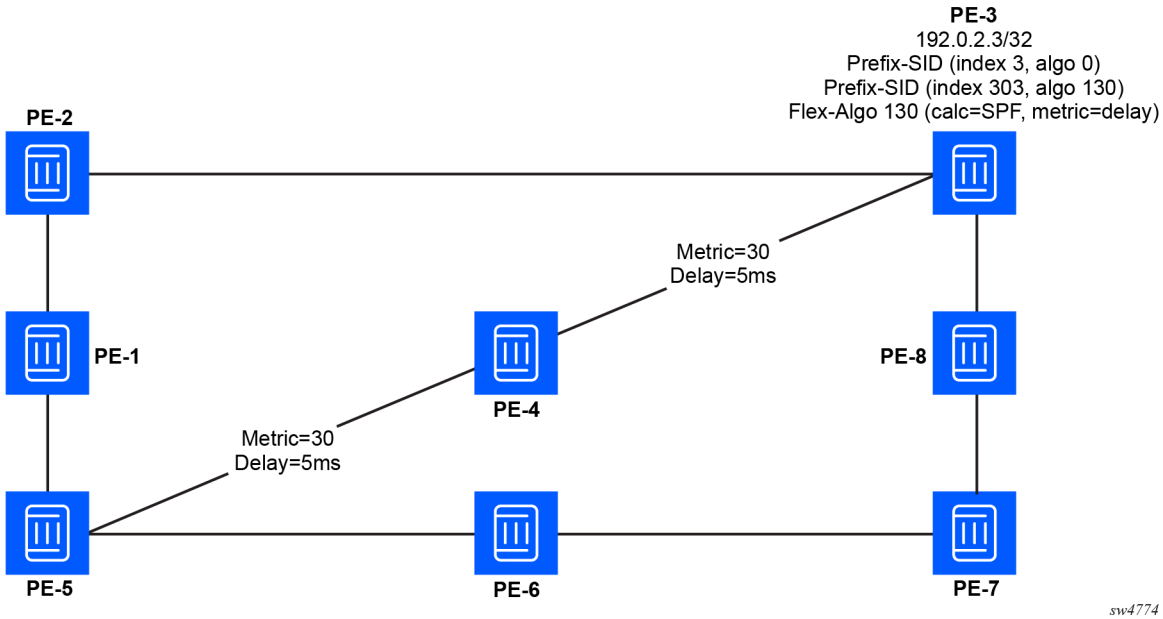
A router may use various algorithms when calculating reachability to other nodes or prefixes attached to those nodes. RFC 8667 — *IS-IS extensions for SR* — describes the use of the SR-Algorithm sub-TLV (carried as part of the Router Capabilities TLV) to advertise the algorithms that the router can support. By default, an SR router will signal support for algorithm 0 (metric-based SPF). To advertise participation for a specific Flex-Algo for SR, the Flex-Algo value must also be advertised in the SR-Algorithm sub-TLV.

When an SR router advertises a Prefix SID, it includes an SR-algorithm, so it is possible to associate a Prefix SID with a specific algorithm. For example, a router may advertise prefix P1 with Prefix SID {index=1, algorithm=0} and prefix P2 with Prefix SID {index=2, algorithm=128}. This indicates to other SR routers that to reach prefix P1, the default metric-based SPF should be used to calculate the best path, and to reach prefix P2, Flex-Algo 128 (and whatever that algorithm dictates) should be used.

Equally, in an SR-MPLS environment with an SR Global Block (SRGB) of {1000-1999}, a router may advertise prefix P1 with Prefix SID {index=1, algorithm=0}, and also Prefix SID {index=101, algorithm=129}. This indicates to other SR routers that when label 1001 is the active label to reach prefix P1, the default metric-based SPF should be used to calculate the best path, and when label 1101 is the active label, Flex-Algo 129 should be used.

The following diagram shows an SR-MPLS domain where all links have metric 10 except for links PE-5-PE-4 and PE-4-PE-3, which both have metric 30. All links have a unidirectional link latency of 10 ms, except for links PE-5-PE-4 and PE-4-PE-3, which both have a unidirectional latency of 5 ms. All routers use an SRGB of {1000-1999}.

Figure 10: Flexible Algorithm example in an SR-MPLS domain



In addition to the default algorithm 0 (metric-based SPF), all routers participate in Flex-Algo 130 with FAD {calc-type=SPF, metric=delay, constraints=none}. Router PE-3 advertises prefix 192.0.2.3/32 with Prefix SID {index=3, algorithm=0} and Prefix SID {index=303, algorithm=130}. Router PE-5 has an SR-TE LSP provisioned with a destination of PE-3 (192.0.2.3) and a top (active) label of 1003. As a result, it

is associated with algorithm 0 and uses the shortest path IGP metric PE-5-PE-1-PE-2-PE-3 to reach its destination. Router PE-5 is also provisioned with a second SR-TE LSP, again with a destination of PE-3 (192.0.2.3), but this time with a top (active) label of 1303. This second LSP is associated with Flex-Algo 130 and uses the shortest path delay metric PE-5-PE-4-PE-3 to reach its destination.

12.4 Configuring the FAD

Procedure

To guarantee loop-free forwarding for paths that are computed for a specific Flex-Algo, all routers configured to participate in that Flex-Algo must agree on the FAD. The agreement ensures that routing loops and inconsistent forwarding behavior is avoided.

Each router that is configured to participate in a specific Flex-Algo must select the FAD based on standardized tie-breaking rules. This ensures consistent FAD selection in cases where different routers advertise different definitions for a specific Flex-Algo. The following tie-breaking rules apply:

- From the FAD advertisements in the area (including both locally generated advertisements and received advertisements), the router selects the one with the highest priority value.
- If there are multiple FAD advertisements with the same priority, the router selects one that originated from the router with the highest system ID.

A router that is not participating in a specific Flex-Algo is allowed to advertise the FAD for that specific Flex-Algo. Any change in the FAD may result in temporary disruption of traffic that is forwarded based on those Flex-Algo paths. The impact is similar to any other event that requires network-wide convergence.

If a node is configured to participate in a Flex-Algo but the selected FAD includes a calculation type, metric type, constraint, flag, or sub-TLV that is not supported by the node, the node stops participation and removes any forwarding state associated with the Flex-Algo.

To configure the FAD, use the **flexible-algorithm flexible-algorithm-definitions flexible-algorithm-definition** command.

Example: Configure FAD

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default flexible-algorithm flexible-algorithm-
definitions
    network-instance default {
        flexible-algorithm {
            flexible-algorithm-definitions {
                flexible-algorithm-definition 128 {
                    admin-state enable
                    metric-type delay
                    priority 237
                    flags-tlv true
                }
            }
        }
    }
```

12.5 Including or excluding extended administrative groups in the FAD

About this task

To apply additional constraints on the Flex-Algo best path calculations, you can assign extended administrative groups to specific links, and then configure the FAD to include or exclude the extended administrative groups as required.

Note: While RFC 9350 directs nodes to conservatively advertise only the extended administrative groups, SR Linux liberally accepts and validates both administrative groups and extended administrative groups.

To configure extended administrative groups, perform the following steps:

Procedure

Step 1. Define the extended administrative group name (color) and bit position using the following command:

flexible-algorithm global-attributes extended-admin-groups group

The bit position value (from 0 to 255) corresponds to one of the positions in the extended admin group bitmask.

Step 2. Assign the extended administrative groups to interfaces using the following command:

flexible-algorithm interface-attributes interface extended-admin-group

The interfaces configured with Flex-Algo attributes cannot be loopback or system interfaces.

Step 3. Configure the FAD to include or exclude extended administrative groups using the following **flexible-algorithm-definition** parameters:

- **include-any** — Includes any links that are associated with at least one of the referenced extended administrative groups in the path calculation.
- **include-all** — Includes only links that are associated with all of the referenced extended administrative groups in the path calculation. Links with only a subset of the specified extended admin-groups are excluded.
- **exclude** — Excludes links that are associated with the referenced extended administrative groups from the path calculation.

Example: Define the extended administrative group name and bit position

```
--{ candidate shared default }--[ ]--
# info with-context network-instance default flexible-algorithm global-attributes
 extended-admin-groups
    network-instance default {
        flexible-algorithm {
            global-attributes {
                extended-admin-groups {
                    group blue {
                        bit-position 3
                    }
                    group green {
                        bit-position 2
                    }
                    group red {
                        bit-position 1
```

```
                    }
                }
            }
        }
    }
```

Example: Assign extended administrative groups to interfaces

```
--{ candidate shared default }--[ ]--
# info with-context network-instance default flexible-algorithm interface-attributes
    network-instance default {
        flexible-algorithm {
            interface-attributes {
                interface ethernet-1/1.1 {
                    extended-admin-group [
                        red
                    ]
                    interface-ref {
                        interface ethernet-1/1
                        subinterface 1
                    }
                }
                interface ethernet-1/2.1 {
                    extended-admin-group [
                        green
                    ]
                    interface-ref {
                        interface ethernet-1/2
                        subinterface 1
                    }
                }
                interface ethernet-1/3.1 {
                    extended-admin-group [
                        blue
                    ]
                    interface-ref {
                        interface ethernet-1/3
                        subinterface 1
                    }
                }
            }
        }
    }
```

Example: Configure the FAD to include and exclude extended administrative groups

```
--{ candidate shared default }--[ ]--
# info with-context network-instance default flexible-algorithm flexible-algorithm-
definitions flexible-algorithm-definition 128
     network-instance default {
        flexible-algorithm {
            flexible-algorithm-definitions {
                flexible-algorithm-definition 128 {
                    admin-state enable
                    metric-type delay
                    priority 237
                    flags-tlv true
                    exclude [
                        red
                    ]
                    include-any [
                        blue
```

```
                        green
                    ]
                }
            }
        }
    }
```

12.6 Configuring IS-IS Flex-Algo advertisement and participation

Procedure

After the FAD is defined, use the **flexible-algorithm-binding** command within the IS-IS instance to advertise the FAD into IS-IS. The Flex-Algo must be assigned a numeric identifier in the range of 128 to 255. The **advertised true** command advertises the locally specified FAD. The **participate true** command configures participation for the specific Flex-Algo and must be enabled on all routers that are part of this Flex-Algo topology.

Finally, LFA may be enabled. If it is, the LFA SPF uses the same Flex-Algo topology as that used to calculate the primary path. Also, LFA settings (such as TI-LFA, Remote-LFA) within a Flex-Algo are inherited from the base IS-IS/LFA configuration.

Example: Configure IS-IS Flex-Algo advertisement and particpation

The following example configures IS-IS to advertise the previously configured FAD 128 and enables the router to participate in the specified algorithm.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance sr-isis-1 segment-
routing flexible-algorithm-binding 128
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    segment-routing {
                        flexible-algorithm-binding 128 {
                            isis-level l1l2
                            advertised true
                            participate true
                            loopfree-alternate true
                        }
                    }
                }
            }
        }
```

12.7 Configuring Flex-Algo prefix node SIDs

Procedure

At each egress node, a Prefix Node-SID must be assigned to each Flex-Algo in use. This is advertised as a Prefix SID sub-TLV that contains (among other things) the algorithm to use to reach the associated Prefix Node-SID. The Node-SID is taken from the generic SRGB; no special or dedicated label space is required.

A prefix node SID (IPv4 or IPv6) must be assigned for each participating Flex-Algo.

The Flex-Algo SIDs are allocated from the label block assigned to segment routing. Configuring a special range is not required.

Example: Configure Flex-Algo prefix node SID

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default segment-routing mpls local-prefix-sid 1 flex-algo 128 ipv4-node-sid
    network-instance default {
        segment-routing {
            mpls {
                local-prefix-sid 1 {
                    flex-algo 128 {
                        ipv4-node-sid {
                            index 101
                        }
                    }
                }
            }
        }
    }
```

12.8 Flex-Algo traffic steering using static routes with indirect next hops

An indirect next-hop refers to a next-hop address that is reachable via an intermediate route rather than being directly connected to the local router. When a static route with an indirect next-hop is configured, the local router performs recursive lookups to determine the appropriate forwarding information to reach the final destination. This process enables the router to forward packets via one or more indirect next-hops that are not directly connected but reachable through existing routes, ensuring correct path resolution.

When combined with Flex-Algo, static routes with indirect next hops provide granular control over traffic flows, allowing you to steer traffic to destinations using adaptable algorithms to enhance overall network efficiency and resilience.

Benefits of Flex-Algo for traffic steering with static routes

When integrated with static routes using indirect next-hops, Flex-Algo enhances traffic engineering capabilities by optimizing path selection, ensuring that traffic is steered efficiently across the network in accordance with predefined constraints and objectives.

The following are some example Flex-Algo use cases:

- Load balancing — Distributes traffic evenly across multiple paths to prevent congestion and optimize resource utilization.
- Latency optimization — Prioritizes paths with lower latency to enhance application performance.
- Cost-based routing — Selects paths based on predefined cost metrics, balancing resource usage and performance.

Without support for Flex-Algo, indirect next-hops are resolved using a best-effort path, which may not provide the required performance for advanced use cases.

Indirect next-hop resolution over an algorithm-aware SR-MPLS tunnel

SR Linux tunnel resolution allows indirect next-hops to be resolved over an SR-MPLS tunnel. Without this capability, indirect next-hops are resolved solely using the IP routing table, without awareness of segment routing constraints.

The SR-MPLS tunnel resolution mechanism is algorithm-aware, allowing indirect next-hops to resolve to a Flex-Algo path with a specific Flex-Algo algorithm. The Flex-Algo path provides enhanced traffic steering based on algorithm-specific constraints.

The SR-MPLS tunnels are assigned at the next-hop **group** and child **nexthop** level.

SR Linux also supports a falllback mechanism for SR-MPLS tunnel resolution. When the desired algorithm-aware SR-MPLS tunnel cannot be resolved, you can designate the default algorithm SR-MPLS tunnel as a fall back.

MPLS-based next-hop resolution is mutually exclusive with other next-hop types

Within the same next-hop-group, indirect MPLS-based next-hop tunnel-resolution is mutually exclusive with other next-hop resolution types. Indirect MPLS next-hops cannot be combined with the following tunnel-resolution next-hop types:

- regular IP next-hops
- direct MPLS next-hops
- GRE tunnel next-hops

Fallback options

When the desired algorithm-aware SR-MPLS tunnel cannot be resolved, you can specify fallback options using the **tunnel-resolution flex-algo-mode [spf-tunnel-fallback | strict | disabled]** command, where:

- **spf-tunnel-fallback**
  Specifies to use the Flex-Algo tunnel when available; otherwise, fall back to the default algorithm SR-MPLS tunnel.
- **strict**
  Enforces strict resolution via the specified Flex-Algo tunnel, rejecting resolution if unavailable.
- **disabled**
  Disables resolution to Flex-Algo tunnels and enforces strict resolution to default SR-MPLS tunnels using algorithm 0. The selection of eligible tunnels is governed by the **allowed-tunnel-types** configuration. If no tunnel types are specified, no tunnel resolution occurs, and recursive resolution falls back to IP next hops.

12.8.1 Configuring an indirect next hop group for Flex-Algo traffic steering

Procedure

To configure a static **next-hop-group** for Flex-Algo traffic steering with an indirect next-hop, use the **next-hop-groups** command to configure the following parameters:

- **tunnel-resolution**
  Specifies the Flex-Algo configuration for indirect static-routes, applied at the next-hop-group or child next-hop level. Also provides a container for the following parameters:
  - **algorithm**
    Specifies the tunnel algorithm (128 to 255).

  - **flex-algo-mode**
    Specifies tunnel fallback options: **spf-tunnel-fallback**, **strict**, or **disabled**.
  - **allowed-tunnel-types**
    Specifies the allowed tunnel types (**sr-isis**).

- **nexthop**
  Specifies an index value for the next-hop and provides a container for the following parameters:
  - **ip-address**
    Specifies the IP address of the indirect next-hop.
  - **resolve true**

    Enables the system to recursively resolve the indirect next-hop address.

    The configuration of **resolve true** is a prerequisite for enabling **tunnel-resolution** to an indirect next-hop over an SR-MPLS tunnel. This applies when resolving the next-hop via Algorithm 0 (default SPF) or a user-defined flexible algorithm. Without **resolve true** enabled, the system does not attempt any resolution and expects the configured **nexthop** IP address to be a directly connected neighbor.
  - **tunnel-resolution**

    When configured for a next-hop-group, the tunnel-resolution configuration is inherited by each child **nexthop** context, unless the child **nexthop** configuration specifies different settings.

Example: Configure Flex-Algo traffic steering using static routes with indirect next hops

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default next-hop-groups group my-flex-group
    network-instance default {
        next-hop-groups {
            group my-flex-group {
                admin-state enable
                tunnel-resolution {
                    algorithm 128
                    flex-algo-mode spf-tunnel-fallback
                    allowed-tunnel-types [
                        sr-isis
                    ]
                }
                nexthop 10 {
                    ip-address 10.20.20.20
                    resolve true
                }
                nexthop 20 {
                    ip-address 10.30.30.30
                    resolve true
                    tunnel-resolution {
                        algorithm 129
                        flex-algo-mode strict
                        allowed-tunnel-types [
                            sr-isis
                        ]
                    }
                }
            }
        }
    }
```

The following table describes the behavior depending on the settings applied to the **resolve**, **algorithm**, **flex-algo-mode**, and **allowed-tunnel-type** parameters.

Table 28: Tunnel resolution behavior

resolve setting	algorithm setting	flex-algo-mode setting	allowed-tunnel-type setting	Tunnel resolution
true	no algorithm configured	**spf-tunnel-fallback**	**sr-isis**	Resolves to algorithm 0 tunnels (default SR-MPLS tunnels).
true	no algorithm configured	**strict**	**sr-isis**	Resolves only to algorithm 0 tunnels, ignoring any Flex-Algo tunnels.
true	no algorithm configured	**disabled**	**sr-isis**	Resolves only to algorithm 0 tunnels, ignoring any Flex-Algo tunnels.
true	no algorithm configured	**spf-tunnel-fallback**, **strict**	None specified	No tunnel resolution; configuration is rejected.
true	no algorithm configured	**disabled**	None specified	Falls back to IP next-hop (base SPF), ignoring all tunnels including Flex-Algo tunnels.
true	**128** to **255**	**spf-tunnel-fallback**	**sr-isis**	Prefers matching to the Flex-Algo tunnel, and falls back to algorithm 0 tunnels if the Flex-Algo tunnel is unavailable.
true	**128** to **255**	**strict**	**sr-isis**	Resolves only to the Flex-Algo tunnel, with no fallback to algorithm 0 tunnels.
true	**128** to **255**	**disabled**	**sr-isis**	Resolves only to algorithm 0 tunnels, ignoring Flex-Algo tunnels.
true	**128** to **255**	**spf-tunnel-fallback**, **strict**	None specified	No tunnel resolution; configuration is rejected.
true	**128** to **255**	**disabled**	None specified	Falls back to IP next-hop (base SPF), ignoring all tunnels including Flex-Algo tunnels.
false	Not applicable	Not applicable	Not applicable	Without **resolve true** enabled, the system does not attempt any resolution and expects the configured next-hop IP address to be a directly connected neighbor.

12.8.2 Associating the Flex-Algo indirect next hop group with a static route

Procedure

To associate the Flex-Algo indirect next-hop configuration with a static route, use the **static-routes** command to specify the target payload destination route, and associate the route with the indirect **static-next-hop-group**.

Example: Configuring static routes

The following example configures a static route to use indirect static next-hop-group **my-flex-group**, defined in the preceding procedure.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default static-routes
    network-instance default {
        static-routes {
            route 10.10.10.0/24 {
                static-next-hop-group my-flex-group
            }
        }
    }
```

12.9 Flex-Algo traffic steering using BGP automated steering

BGP automated steering using Flex-Algo refers to the use of adaptive, programmable algorithms within BGP routing processes to automatically manage and optimize the selection of routes. It leverages real-time performance data and adaptable algorithms to make informed routing decisions without manual intervention and enhance network performance, resilience, and operational efficiency.

Some example use cases include:

- Traffic engineering – Optimizes the distribution of traffic across multiple links to prevent congestion and balance load.
- Disaster recovery – Automatically reroutes traffic in the event of a link or node failure to maintain network connectivity.
- Quality of service (QoS) management – Steers traffic to routes that meet specific QoS requirements, such as low latency for real-time applications.
- Cost optimization – Selects routes that minimize operational costs by favoring lower-cost links while maintaining performance standards.

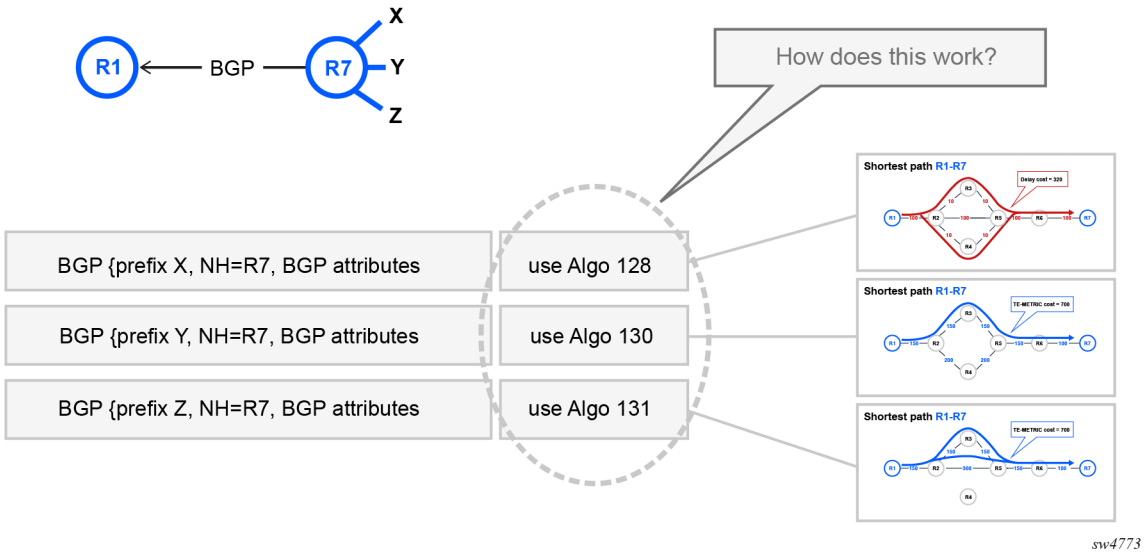
BGP automated steering operational model

At a high level, automated steering using BGP requires two operational steps:

1. Tagging BGP NLRI (unicast, IP-VPN, EVPN, or BGP-LU)
2. Steering NLRI payload traffic based on the tag

The high level principle is shown in the following figure.

Figure 11: Flex-Algo BGP automated steering



Tagging BGP NLRI

Tagging BGP NLRI typically consists of assigning the NLRI an extended community using an import route policy, but the tag can be any other attribute that a BGP route policy can match. Standard route policy commands are sufficient to add the required tag for FlexAlgo traffic steering.

Steering NLRI payload traffic based on the tag

To steer NLRI payload traffic, you must configure import route policy statements to match on the extended community and instruct the router to resolve the BGP next-hop using a specific Flex-Algo tunnel. At a high level, the steps are as follows:

1. Use a route policy match statement to match the desired BGP NLRI and extended community.
2. Define a route policy action to specify the algorithm associated with the BGP NLRI. Automated traffic steering requires a **set-flex-algo** action in the route policy specifying the desired algorithm (128-255) as the next-hop for the matching traffic.
3. Enable BGP Flex-Algo aware next-hop resolution by specifying **flex-algo** as a selection attribute for tunnel resolution, which binds Flex-Algo aware LSPs to the BGP next-hop.

Tunnel-resolution mode and flex-algo parameter settings

When Flex-Algo is assigned using an import route policy, the Flex-Algo behavior depends on **tunnel-resolution mode** and **flex-algo mandatory** parameter settings, as described in the following table.

Table 29: Flex-Algo operation by tunnel-resolution and flex-algo parameter setting

Parameter settings		Result
tunnel-resolution	**flex-algo mandatory**	
require	**true**	The IGP computes Flex-Algo MPLS segment routing LSP resolution to resolve the next-hop. If no Flex-Algo aware LSP exists, the next-hop resolution fails.
	false	Flex-Algo tunnel resolution is disabled and the specified Flex-Algo is ignored. Instead the system performs next-hop resolution to the best tunnel (segment routing, LDP, and so on) based on configured tunnel preference. If no applicable tunnels exist, then the next-hop is not resolved.
prefer	**true**	The IGP computes Flex-Algo MPLS segment routing LSP resolution to resolve the next-hop. If no Flex-Algo aware LSP exists, the next-hop resolution fails. (IP route resolution for Flex-Algo aware next-hops is not supported.)
	false	Flex-Algo tunnel resolution is disabled and the specified Flex-Algo is ignored. Instead the system performs next-hop resolution to the best tunnel (segment routing, LDP, and so on) based on configured tunnel preference. If no applicable tunnels exist, then the next-hop is not resolved.
disabled	**true**	With **flex-algo mandatory true**, any resolution must be to an algorithm aware tunnel; however, the **tunnel-resolution mode disabled** indicates that the IP RTM must be used (no tunnels are allowed). If Flex-Algo is assigned through route policy in this case, it is considered an incorrect configuration. Because the IP RTM is not algorithm-aware, the next-hop is unresolved.
	false	Flex-Algo tunnel resolution is disabled and the specified Flex-Algo is ignored. Instead next-hop resolution is resolved to the best IP route entry in the IP routing table. If it is not resolved by the IP routing table, then the next-hop remains unresolved.

Note: When **flex-algo mandatory** is set to **true**, if no Flex-Algo is assigned using an import route policy, it is assumed that no Flex-Algo constraints need to be applied. In this case, the BGP next-hop resolves in the same manner as when **flex-algo mandatory** is set to **false**.

Supported NLRI

BGP based FlexAlgo automated steering supports next-hop resolution for the following NLRI:

- Unlabeled IP routes
- IP-VPN and EVPN-MPLS routes imported into IP VRFs (non ABR/ASBR)
- IP-VPN and EVPN-MPLS routes at an ABR/ASBR
- Labeled-unicast (BGP-LU) routes

Tunnel-resolution mode setting not supported with BGP-LU

The BGP-LU address-family does not support a **tunnel-resolution** mode setting. Instead, BGP-LU has a **route-resolution** container that provides equivalent functionality.

- When **route-resolution admin-state** is **disabled** (the default), the implicit **tunnel-resolution mode** is **require**.
- When **route-resolution admin-state** is **enabled**, the implicit **tunnel-resolution mode** is **prefer**.

BGP Flex-Algo aware resolution matrix

The following definitions describe the headings included in the following table:

- **Allowed tunnels**
  The list of protocols that can be used to resolve tunnels.
- **Algorithm for Next-Hop Resolution**
  When a BGP NLRI is received, an ingress route policy can assign a flexible algorithm to resolve the next hop via a Flex-Algo tunnel. If no Flex-Algo is specified, the default algorithm 0 is used.
- **Existing SR-ISIS Tunnels to Next Hop**
  The list of segment routing tunnels, by algorithm, available toward the BGP next hop.
- **mandatory**
  The boolean **flex-algo mandatory** as described in the preceding section.
- **mode**
  The configuration option **tunnel-resolution mode** as described in the preceding section.
- **Final Resolution Expected**
  The expected outcome when the configured conditions from prior configuration and state options are applied.
- **Notes**
  Any specific additional information as needed.

Table 30: BGP Flex-Algo aware resolution

allowed tunnels	Algorithm for Next-Hop Resolution(by route policy)	Existing SR-ISIS Tunnels to Next Hop	mandatory	mode	Final resolution Expected	Notes
LDP, SR-ISIS	129	129, 0	TRUE	prefer	SR-ISIS <algo 129>	—
	not set	129, 0	TRUE	prefer	LDP, SR-ISIS <algo 0>, RTM	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints are applied and BGP next-hop recursion follows the standard behavior equivalent to when **flex-algo mandatory false** is set.

allowed tunnels	Algorithm for Next-Hop Resolution(by route policy)	Existing SR-ISIS Tunnels to Next Hop	mandatory	mode	Final resolution Expected	Notes
	129	129, 0	FALSE	prefer	LDP, SR-ISIS <algo 0>, RTM	Resolution is based on the configured tunnel preference. Flex-Algo tunnels are ignored.
	not set	129, 0	FALSE	prefer	LDP, SR-ISIS <algo 0>, RTM	Resolution is based on the configured tunnel preference. Flex-Algo tunnels are ignored.
LDP, SR-ISIS	129	0	TRUE	prefer	unresolved	—
	not set	0	TRUE	prefer	LDP, SR-ISIS <algo 0>, RTM	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints need to be applied and BGP next-hop recursion follows the standard behavior equivalent to when **flex-algo mandatory false** is set.
	129	0	FALSE	prefer	LDP, SR-ISIS <algo 0>, RTM	Resolution is based on configured tunnel preference. Flex-Algo tunnels are ignored.
	not set	0	FALSE	prefer	LDP, SR-ISIS <algo 0>, RTM	Resolution is based on configured tunnel preference. Flex-Algo tunnels are ignored.
LDP, SR-ISIS	129	129, 0	TRUE	require	SR-ISIS <algo 129>	—
	not set	129, 0	TRUE	require	LDP, SR-ISIS <algo 0>	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints need to be applied and BGP next-hop recursion follows the standard behavior equivalent to when

allowed tunnels	Algorithm for Next-Hop Resolution(by route policy)	Existing SR-ISIS Tunnels to Next Hop	mandatory	mode	Final resolution Expected	Notes
						flex-algo mandatory false is set.
	129	129, 0	FALSE	require	LDP, SR-ISIS <algo 0>	Resolution is based upon configured tunnel-preference. Fallback to RTM recursion not allowed. Flexible algorithm tunnels are ignored.
	not set	129, 0	FALSE	require	LDP, SR-ISIS <algo 0>	Resolution is based upon configured tunnel-preference. Fallback to RTM recursion not allowed. Flexible algorithm tunnels are ignored.
LDP, SR-ISIS	129	0	TRUE	require	unresolved	—
	not set	0	TRUE	require	LDP, SR-ISIS <algo 0>	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints need to be applied and BGP next-hop recursion follows the standard behavior equivalent to when **flex-algo mandatory false** is set.
	129	0	FALSE	require	LDP, SR-ISIS <algo 0>	Resolution is based on configured tunnel preference. Fallback to RTM recursion is not allowed.
	not set	0	FALSE	require	LDP, SR-ISIS <algo 0>, LDP	Resolution is based on configured tunnel preference. Fallback to RTM recursion is not allowed.
LDP, SR-ISIS	129	129, 0	TRUE	disabled	unresolved	—

allowed tunnels	Algorithm for Next-Hop Resolution(by route policy)	Existing SR-ISIS Tunnels to Next Hop	mandatory	mode	Final resolution Expected	Notes
	not set	129, 0	TRUE	disabled	RTM (plane ISIS IP routing)	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints need to be applied and BGP next-hop recursion follows the standard behavior equivalent to when **flex-algo mandatory false** is set.
	129	129, 0	FALSE	disabled	RTM (plane ISIS IP routing)	—
	not set	129, 0	FALSE	disabled	RTM (plane ISIS IP routing)	—
LDP, SR-ISIS	129	0	TRUE	disabled	unresolved	—
	not set	0	TRUE	disabled	RTM (plane ISIS IP routing)	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints need to be applied and BGP next-hop recursion follows the standard behavior equivalent to when **flex-algo mandatory false** is set.
	129	0	FALSE	disabled	RTM (plane ISIS IP routing)	—
	not set	0	FALSE	disabled	RTM (plane ISIS IP routing)	—
empty	129	129, 0	FALSE	require	unresolved	—
	not set	129, 0	FALSE	require	unresolved	—
	129	129, 0	TRUE	prefer	unresolved	—
	not set	129, 0	FALSE	require	unresolved	—

allowed tunnels	Algorithm for Next-Hop Resolution(by route policy)	Existing SR-ISIS Tunnels to Next Hop	mandatory	mode	Final resolution Expected	Notes
	N/A	129, 0	N/A	disabled	RTM (plane ISIS IP routing)	—
LDP	N/A	129, 0	FALSE	require	LDP	—
	129	129, 0	TRUE	prefer	unresolved	—
	not set	129, 0	TRUE	prefer	LDP	If no Flex-Algo is assigned through route policy, no Flex-Algo constraints need to be applied and BGP next-hop recursion follows the standard behavior equivalent to when **flex-algo mandatory false** is set.
	N/A	129, 0	FALSE	prefer	LDP	—

Note: The SR Linux fib_mgr does not support negative matching to allow exclusion of specific algorithms (for example, **sr-isis algo-129**) while permitting all others. However, when **flex-algo mandatory** is set to **false** and a route policy assigns a Flex-Algo to an NLRI route, the system applies alternative resolution logic and prioritizes first toward an SR-ISIS (algorithm 0) tunnel before trying other tunnels (LDP for example) or RTM next hops.

12.9.1 Configuring a routing policy for Flex Algo traffic steering with BGP

Procedure

Use the **routing-policy** command to define a policy that matches on an **extended-community** and defines the value of the Flex-Algo tunnel to use for traffic steering.

Example:

```
-{ candidate shared default }--[  ]--
# info with-context routing-policy
    routing-policy {
        policy flex-policy {
            statement 100 {
                match {
                    bgp {
                        extended-community {
                            extended-community-set e1
                            match-set-options any
                        }
                    }
                }
```

```
                action {
                    bgp {
                        next-hop-resolution {
                            set-flex-algo 128
                        }
                    }
                }
            }
        }
    }
```

12.9.2 Configuring Flex-Algo traffic steering using BGP automated steering

Procedure

Use the **tunnel-resolution** command to define the Flex-Algo traffic steering settings for BGP. The **tunnel-resolution mode** must be set to **require** or **prefer** and the **tunnel-resolution allowed-tunnel-types** must specify **sr-isis**; otherwise, the system applies the traditional resolution logic using the tunnel table or route table and the Flex-Algo configuration is ignored.

Example: Configuring Flex-Algo traffic steering using BGP automated steering (non-BGP-LU)

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                admin-state enable
                import-policy [
                    flex-policy
                ]
                afi-safi ipv4-unicast {
                    ipv4-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                tunnel-resolution {
                                    mode prefer
                                    allowed-tunnel-types [
                                        sr-isis
                                    ]
                                    selection-attributes {
                                        tag {
                                            mandatory true
                                        }
                                        flex-algo {
                                            mandatory true
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Configuring Flex-Algo traffic steering using BGP automated steering (BGP-LU)

The BGP-LU address-family does not support a **tunnel-resolution mode** command. Instead, the implicit tunnel-resolution mode for BGP-LU routes is determined by the **route-resolution admin-state** setting, as follows:

- **route-resolution admin-state disable** (default) – equivalent to an implicit **tunnel-resolution mode** of **require**
- **route-resolution admin-state enable** – equivalent to an implicit **tunnel-resolution mode** of **prefer**

The following example shows a BGP-LU next-hop with **route-resolution** enabled, providing the equivalent to a **tunnel-resolution mode** of **require**.

```
--{ candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                admin-state enable
                import-policy [
                    flex-policy
                ]
                afi-safi ipv4-labeled-unicast {
                    ipv4-labeled-unicast {
                        next-hop-resolution {
                            ipv4-next-hops {
                                route-resolution {
                                    admin-state enable
                                }
                                tunnel-resolution {
                                    allowed-tunnel-types [
                                        sr-isis
                                    ]
                                    selection-attributes {
                                        tag {
                                            mandatory true
                                        }
                                        flex-algo {
                                            mandatory true
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

12.10 Delay normalization with Flex-Algo

Interface delay normalization is a network engineering method used to address and manage the different delay times (latency) experienced across a network. Within Flex-Algo, delay normalization provides a method to adjust and standardize delay metrics across different network segments, ensuring consistent and optimized routing decisions. Delay normalization can significantly improve the impact of delay-based

routing decisions, leading to enhanced network reliability, optimized latency, and improved quality of service (QoS).

Delay normalization builds on the principle of reducing the granularity of advertised delay. By default, the measured delay tolerance is expressed in 1 microsecond units whether the delay is small or large. With delay normalization, the dynamically measured delay tolerance is artificially changed by the Interior Gateway Protocol (IGP).

Understanding network delays

In a network, data packets often experience different delay times as they travel from their source to their destination. These delays can be caused by various factors, including:

- physical distance
- number of hops (intermediate devices like routers and switches packets pass through)
- network traffic load
- processing speed of the devices

IS-IS can use these characteristics as metrics for Flex-Algo computation. One of the key use cases for Flex-Algo technology is low-latency routing using dynamic delay measurement.

Measuring delay

The first step in interface delay normalization is to measure the delay experienced at various points or interfaces in the network. This measurement is typically done using tools and protocols designed to measure round-trip times (RTT) or one-way delay times. Nokia SR Linux typically deploys Two-Way Active Measurement Protocol (TWAMP) to measure unidirectional interface delay. Delay is measured in microseconds (µsec). However, if the Flex-Algo topology computation uses raw delay values as link metrics, minor differences in link delay can prevent the use of valid ECMP routes.

Normalizing delays

After the delay measurements are obtained, the delay normalization process computes a normalized delay value to use as the metric. This process involves adjusting certain parameters or configurations in the network to make the delay more uniform or predictable across different segments of the network. This normalized value is then advertised through IS-IS using Flex-Algo.

When delay is dynamically measured using TWAMP, mechanisms are in place to avoid frequent updates of the measured interface delay by the IGP. This is achieved by enforcing either a percentage difference or an absolute value difference. The dynamic measured dampening causes the IGP to reduce the frequency of measured delay updates, thereby stabilizing most observed delay measurement dynamics.

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

12.10.1 Delay normalization calculation

The normalized delay is calculated as follows:

Normalized delay = INTEGER DIVISION (measured delay / delay-tolerance-interval) × delay-tolerance-interval + minimum-delay

This calculation ensures that small variations in delay are smoothed out, improving the stability of the delay metric used in IGP computations.

The following table describes the normalized delay calculation using multiple examples.

Table 31: Normalized delay calculation

Measured delay (µsec)	Delay tolerance interval (µsec)	Minimum delay (µsec)	Calculated delay	Resulting normalized delay (µsec)
2	11	5	2 / 11 = 0	0 × 11 + 5 = 5
10	11	5	10 / 11 = 0	0 × 11 + 5 = 5
11	11	5	11 / 11 = 1	1 × 11 + 5 = 16
12	11	5	12 / 11 = 1	1 × 11 + 5 = 16
20	11	5	20 / 11 = 1	1 × 11 + 5 = 16
22	11	5	22 / 11 = 2	2 × 11 + 5 = 27
32	11	5	32 / 11 = 2	2 × 11 + 5 = 27
33	11	5	33 / 11 = 3	3 × 11 + 5 = 38
100	11	5	100 / 11 = 9	9 × 11 + 5 = 104

12.10.2 Configuring delay normalization

Procedure

To configure delay normalization for an IS-IS interface, use the **delay normalization admin-state enable** command.

Example: Configure IS-IS interface delay normalization

```
--{ candidate shared default }--[ ]--
# info with-context network-instance default protocols isis instance sr-isis-1 interface
 ethernet-1/1.1 delay normalization
    network-instance default {
        protocols {
            isis {
                instance sr-isis-1 {
                    interface ethernet-1/1.1 {
                        delay {
                            normalization {
                                admin-state enable
                                minimum-delay 1
                                delay-tolerance-interval 10
                            }
                        }
                    }
                }
            }
        }
    }
```

13 Reporting resource exhaustion per-stage

Note:
This feature is currently supported exclusively on the 7250 IXR Gen 2c+ platform.

The 7250 IXR Gen 2c+ platform provides direct visibility into datapath tables, allowing for monitoring of SR-MPLS resource usage. It offers detailed insights into resource availability and usage for tunnels and their associated protection mechanisms, such as FRR (Fast Reroute) and LFA (Loop-Free Alternate).

SR-MPLS and BGP labeled prefix tunnels are programmed into separate tables within the datapath. Each MPLS or Labeled Unicast transport tunnel (NHLFE/segment list) must be programmed in the appropriate table. TE policy tunnels may require entries to be programmed across multiple tables simultaneously.

Based on the number of labels associated with TE policy segment list, labels are assigned into different phases, and resource exhaustion in any phase leads to TE-policy installation failure. For example, the outermost labels 1 and 2 are assigned to one phase, while labels 3 and 4 are assigned to another. In this setup, a TE policy segment list requiring eight labels would fail to install if any of the phases ran out of free Egress Encapsulation DataBase (EEDB) entries. In addition, the LDP, SR-ISIS, and associated FRR protection mechanisms are included in phase 7, and finally, ti-LFA is included in phase 6.

The **asic** container within the **platform.resource-monitoring.datapath** context is used to set the threshold for monitoring the datapath resources. For a configuration example, see Configuring thresholds for datapath resources.

13.1 Configuring thresholds for datapath resources

Procedure

SR Linux allows you to set thresholds for monitoring the datapath resources.
For each **platform.resource-monitoring.datapath.asic** resource, you can specify two threshold values:

- `upper-threshold-set`: a percentage value (from 0-100) that defines the threshold at which the resource is considered to be in a warning state. When the resource utilization exceeds this threshold, an alert is triggered and a WARNING log is generated.
- `upper-threshold-clear`: a percentage value (from 0-100) that defines the threshold at which the resource is considered to be back in a normal state. When the resource utilization falls below this threshold, the alert is cleared and a NOTICE log is generated.

Example

The following example configures the threshold to monitor the phase-3 Egress Encapsulation Data Base (EEDB) resource usage (**phase-3-type-1-eedb-entries**) for MPLS. When the number of **phase-3-type-1-eedb-entries** entries exceeds 100, a WARNING log is generated, and when it falls below 50, a NOTICE log is generated.

```
--{ + candidate shared default }--[ ]--
# info with-context platform resource-monitoring
    platform {
        resource-monitoring {
            datapath {
```

```
                asic {
                    resource phase-3-type-1-eedb-entries {
                        upper-threshold-set 100
                        upper-threshold-clear 50
                    }
                }
            }
        }
    }
```

14 Segment routing with IPv6 data plane (SRv6)

Note:
SRv6 is currently supported exclusively on 7730 SXR platforms.

Segment routing steers packets by encoding the packet-processing instructions for each intermediate and destination router directly in the packet header.

The datapath pushes a list of instructions, in the form of Segment Identifiers (SIDs), onto the packet to forward the payload directly to the destination using the shortest path, or using source routing using one or more transit routers. Each router that terminates a SID in the segment list executes the instructions associated with that SID.

SRv6 is an IPv6-based segment routing approach that introduces the Segment Routing Header (SRH) as a new IPv6 extension header.

The following table highlights the differences between SR-MPLS and SRv6:

Table 32: Differences between SR-MPLS and SRv6

Feature	SR-MPLS	SRv6
SID encoding	32-bit MPLS label, see Datapath programming by SID type	128-bit IPv6 address
Header	MPLS label stack (no new IPv6 extension header)	Segment Routing Header (SRH) – a new IPv6 extension header
Forwarding plane support	No changes needed for IPv4 or IPv6 forwarding planes (uses existing MPLS forwarding). See Datapath programming by SID type	Requires forwarding plane support to parse and act on the SRH
Tunnel destination	Works for both IPv4 and IPv6 destinations	Works only for IPv6 destinations
Typical use case	Legacy MPLS networks, mixed IPv4/IPv6 environments	Pure IPv6 networks requiring segment routing capabilities

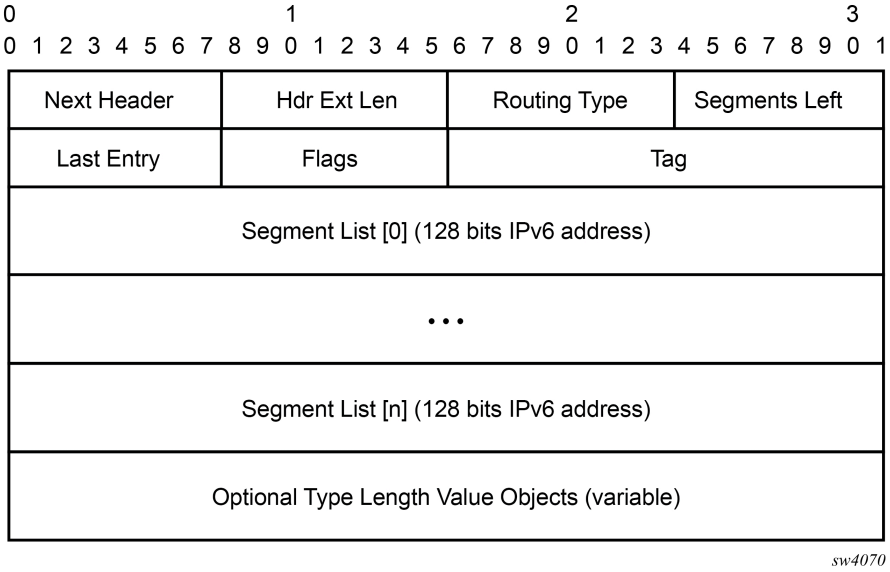
14.1 SRv6 Segment Routing Header (SRH)

SRv6 provides more than just IPv6 transport with shortest path and source-routing capabilities; it provides a framework for IPv6 network programmability that takes advantage of the large IPv6 address space.

In shortest path routing, the destination SID is encoded in the Destination Address (DA) field of the outer IPv6 header. In source routing, the SIDs of the nodes the packet must traverse are encoded as a SID list in a Segment Routing Header (SRH), which is a new type of routing header compliant with RFC 8200. The next SID in a segment list to which the packet is to be forwarded is copied from the SRH into the DA field of the outer IPv6 header.

The following figure shows the SRv6 SRH format and fields (excerpt from RFC 8986).

Figure 12: SRv6 SRH format and fields



The following table lists the SRv6 field descriptions.

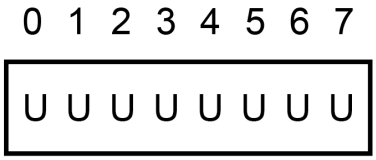
Table 33: SRv6 field descriptions

Field name	Description
Next Header	Defined in RFC 8200, Section 4.4
Hdr Ext Len	Defined in RFC 8200, Section 4.4
Routing Type	Defined in RFC 8200, Section 4.4
Segments Left	Defined in RFC 8200, Section 4.4
Last Entry	Contains the index (zero based), in the Segment List, of the last element of the Segment List
Flags	RFC 8754, Section 8.1 creates an IANA registry for new flags to be defined.
Tag	The Tag field is used to mark a packet as part of a class or group of packets; for example, packets sharing the same set of properties. When the Tag field is not used at the source, it must be set to zero on transmission. When the Tag field is not used during SRH processing, it is ignored. The Tag field is not used when processing the SID, as defined in RFC 8754, Section 4.3.1. It may be used when processing other SIDs that are not defined in this document. The allocation and use of tags are outside the scope of this document.

Field name	Description
Segment List[0..n]	128-bit IPv6 addresses representing the nth segment in the segment list. The Segment List is encoded starting from the last segment of the SR policy. That is, the first element of the segment list (Segment List[0]) contains the last segment of the SR policy, the second element contains the penultimate segment of the SR policy, and so on.
TLV	Type Length Value (TLV); see RFC 8754 Section 2.1, and SRv6 SID encoding.

The following flags are defined for the SRv6 SRH Flags field.

Figure 13: 8-bits of flags



sw4071

The following table lists the SRH flag descriptions.

Table 34: Flag descriptions

Flag	Description
U	Unused and for future use. Must be 0 on transmission and ignored on receipt.

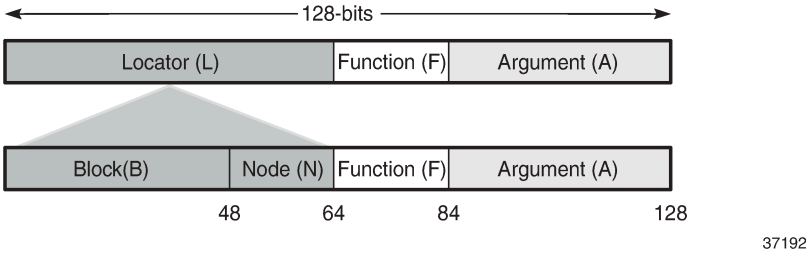
14.2 SRv6 SID encoding

An SRv6 SID is a routable IPv6 prefix when set as the IPv6 header DA.

Note:
IPv6 router interface addresses are not SRv6 SIDs.

The following figure shows that the 128-bit address of an SRv6 SID is split into three constituent parts: locator, function, argument.

Figure 14: SRv6 SID encoding



- The locator is a summary IPv6 prefix for a set of SIDs instantiated on an SRv6-capable router. The locator:
  - must be explicitly configured
  - is advertised using IS-IS
  - can be associated with a topology and/or Flex-Algorithm
  - provides reachability to all SIDs originated by a router if the locator part of the SRv6 SID is routable
  - comprises the L most significant bits of the SID, with L ranging from 4 to 96 bits
  - has format B:N
    - All routers in a domain have the same block address B.
    - Each router in the domain has its own node-specific address N.
- The function is an opaque identification of a local behavior bound to the segment, as described in RFC 8986. The following table lists the supported SRv6 endpoint behaviors.

Table 35: SRv6 endpoint behaviors supported

Function name	Role or behavior	Description
End	Endpoint	Equivalent to a node SID
End.X	Endpoint with an L3 cross-connect (X-connect)	Equivalent to an adjacency SID
LAN-End.X	Endpoint with an L3 cross-connect (X-connect)	Equivalent to an adjacency SID associated with a broadcast interface
End.DT4	De-encapsulate and perform an IPv4 table lookup	– IP-VRF table lookup: per-VRF SID for the VPN-IPv4 address family – Prefix lookup in the global IPv4 routing table
End.DT6	De-encapsulate and perform an IPv6 table lookup	– IP-VRF table lookup: per-VRF SID for the VPN-IPv6 address family

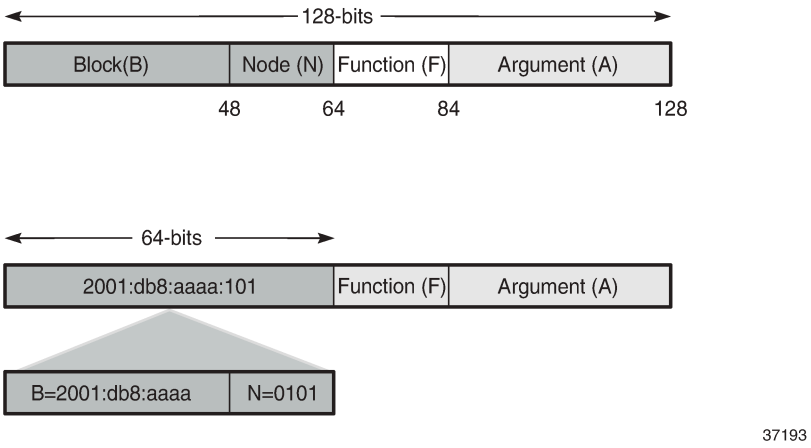
Function name	Role or behavior	Description
		– Prefix lookup in the global IPv6 routing table
End.DT46	De-encapsulate and perform IPv4 and IPv6 table lookups	– IP-VRF table lookup: both IPv4 and IPv6 - equivalent to per-VRF label – VPN-IPv4 and VPN-IPv6 routes are advertised with a single label in the same VRF

- The argument, which is not a configurable field is set to all zeros.

The following figure shows an example of an SRv6 SID with the following:

- B = 48 bits
- N = 16 bits
- L = B + N = 48 bits + 16 bits = 64 bits
- F = 20 bits
- The remaining 44 bits (A) are set to zero.

Figure 15: SRv6 SID encoding example



The /64 locator part for a set of routers in a routing domain consists of:

- a common 48-bit block, for example, 2001:db8:aaaa::/48
- a unique 16-bit node identifier allocated in the range from 0000 to ffff

Some examples:

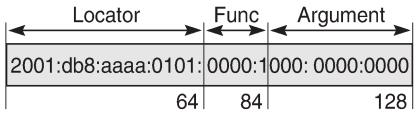
- locator for PE-1 = 2001:db8:aaaa:101::/64
- locator for PE-2 = 2001:db8:aaaa:102::/64
- locator for PE-3 = 2001:db8:aaaa:103::/64

The local router installs the locator in its IPv6 route table and FIB. The locator prefix is advertised in IS-IS in the SRv6 locator sub-TLV. Each remote router populates its route table and FIB with the locator prefixes, including the tunneled next-hop to the originating router.

The function field has a configurable length, ranging from 20 to 96 bits. By default, the function field has 20 bits. The function field is used to assign End and End.X SIDs, which are used by remote routers to create repair tunnels for remote and topology-independent loopfree-alternate (RLFA and TI-LFA) backup paths.

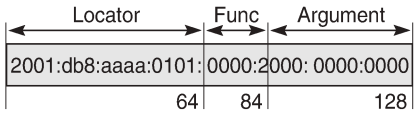
- An End function is statically configured in SR Linux:
  - By default, the number of static functions is 1.
  - For example, the End function with value 1 in the 20-bit format is represented as 00001 in hexadecimal, followed by the zeros of the argument field.
  - The End SID (node SID) for PE-1 equals 2001:db8:aaaa:101:0:1000::/128, as shown in the following figure.

    Figure 16: End SID for PE-1



- The End.X function can be statically configured or automatically assigned by the system.
  - In case of static configuration, the number of static functions must be increased.
  - For the function with value 2 in a function field of 20 bits, the corresponding hexadecimal pattern is 00002, followed by the zeros of the argument field.
  - The End.X SID (adjacency SID) for PE-1 equals 2001:db8:aaaa:101:0:2000::/128, as shown in the following figure.

    Figure 17: End.X SID for PE-1

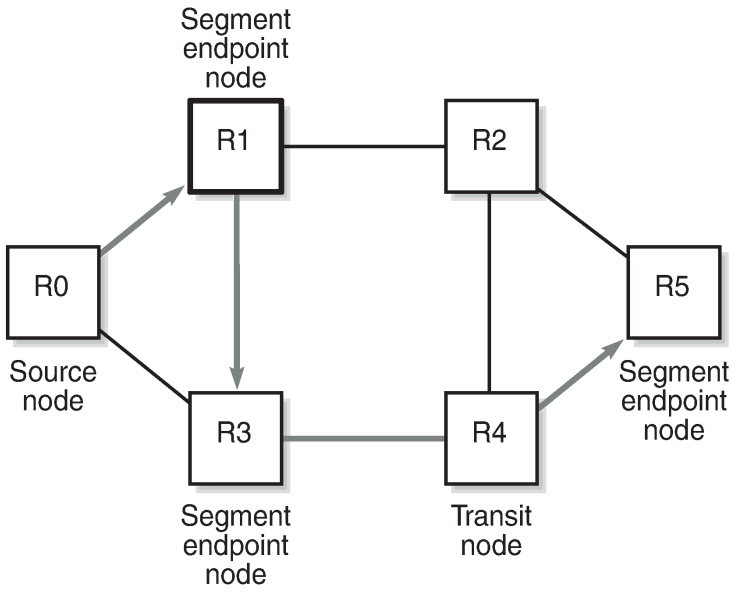


14.3 SRv6 node types

Different SR node types are defined: source node, transit node, and segment endpoint node.

The following figure shows the SR node types for an SRv6 packet flow from R0 to R5 via hops R1, R3, and R5.

Figure 18: SRv6 node types



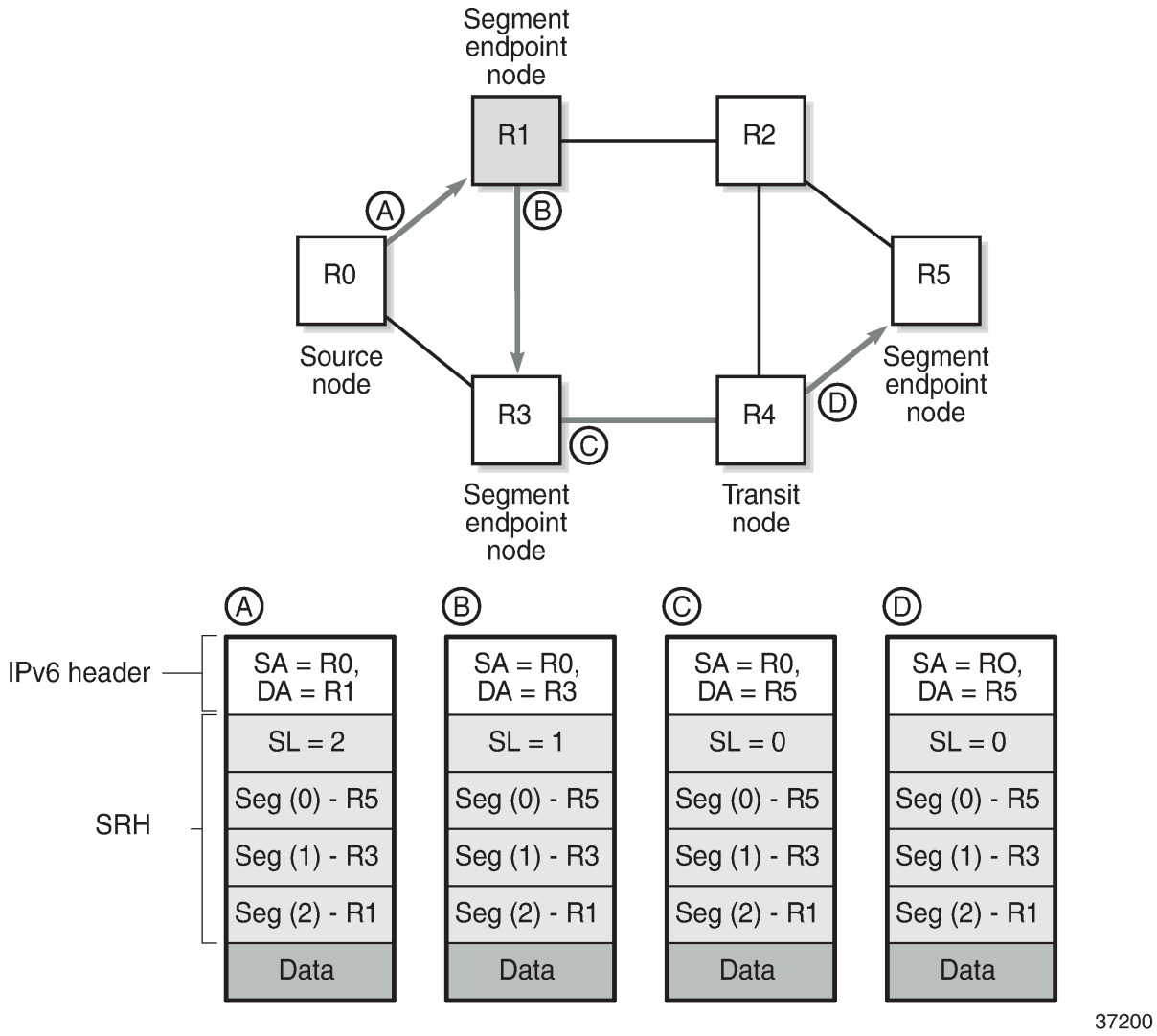
The intermediate hops R1, R3, and R5 are programmed in the segment list of the SRH.

The SRv6 node types defined in RFC 8754 are:

- SR source node
  - Any node that originates an IPv6 packet with a segment (that is, an SRv6 SID) in the DA field of the IPv6 header.
  - The IPv6 packet leaving the SR source node may or may not contain an SRH. This includes either:
    - a host originating an IPv6 packet
    - an SR domain ingress router encapsulating a received packet in an outer IPv6 header, followed by an optional SRH
  - In this example, R0 acts as an SR source node and includes an SRH containing a segment list.
- SR transit node
  - Any node forwarding an IPv6 packet where the DA of the packet is not locally configured as a segment or a local interface. A transit node does not need to be capable of processing a segment or SRH.
  - In this example, R4 acts as an SR transit node. It forwards the SRv6 packet without processing the SRH.
- SR segment endpoint node
  - Any node receiving an IPv6 packet where the DA of that packet is locally configured as a segment or local interface.
  - In this example, R1, R3, and R5 are SR segment endpoint nodes. These nodes interrogate the SRH as part of packet processing.

The following figure shows the data forwarding of SRv6 encapsulated packets using SRv6 SIDs at R0 and R1.

Figure 19: Data forwarding of SRv6 encapsulated packets using SRv6 SIDs



Source node R0 tunnels an SRv6 packet to destination R5, segment (0), in the SRH.

- The segment list contains SRv6 SIDs associated with each hop, such as the End SID. The first segment endpoint is the last segment in the list, segment (2) in the example. The Segments Left (SL) field is set to a value matching the highest segment list number (2).
- SRH is only used by routers where the DA is equal to a local address. The IPv6 source address is set to the local IPv6 address of R0. The IPv6 DA in the IPv6 header is set to the segment list entry indexed in the SL field; in this case, R1.
- The packet is forwarded to R1.

At R1, the incoming packet has the IPv6 DA matching R1.

- R1 removes the IPv6 header and processes the SRH. The SL is decremented to SL 1, which corresponds to segment (1) = R3.
- R1 adds an IPv6 header with DA equal to the SID for R3.
- R1 forwards the packet to R3.

At R3, the incoming packet has the IPv6 DA matching R3.

- R3 removes the IPv6 header and processes the SRH. The SL is decremented to SL 0, which corresponds to segment (0) = R5.
- R3 adds an IPv6 header with DA equal to the SID for R5.
- R3 forwards the packet to R5.

At R4, the incoming packet has the IPv6 DA matching R5, so the packet is forwarded to R5 without processing the SRH header and without changing the IPv6 DA.

At R5, the incoming packet has the IPv6 DA matching R5, so the IPv6 header is removed and the SRH header is processed. The SL value 0 cannot be decreased anymore, so R5 removes the SRH and the packet is sent for further processing, for example, to a particular IP-VRF.

Note:
The IPv6 SID at segment (0) may contain an opaque behavior value (function) that indicates to the destination node that further processing is required, such as a IP-VRF table lookup.

14.4 SRH processing modes

SR Linux supports the following SRH processing modes at the end of the SRv6 tunnel:

- Ultimate SRH Pop (USP), where the ultimate SR segment endpoint node processes and removes the SRH
- Penultimate SRH Pop (PSP), where the penultimate SR segment endpoint node processes and removes the SRH
- Ultimate Segment Decapsulation (USD), where the ultimate SR segment endpoint decapsulates the entire outer IPv6 header and all extension headers (including SRH) at the final node to forward the inner payload.

14.5 SRv6 Micro-SID (uSID)

SR Linux supports micro-segment SRv6. The system supports concurrent operation in both modes (regular and micro) on the same platform, but requires that the SIDs for each type come from different SID blocks.

Micro-segment SRv6 provides the same functionality as regular SRv6 but uses 16-bit SIDs, known as micro-SIDs, instead of 128-bit SIDs.

Micro-segment (uSID) encodes multiple short instructions (micro-SIDs) into a single 128-bit IPv6 address container called a uSID Carrier.

Micro-SIDs follow the same general structure as regular SIDs (LOCATOR:FUNCTION:ARGUMENT), but differ in key aspects described in the following table.

Table 36: Differences between full-segment (SID) and micro-segment (uSID)

Features	full-segment (SID)	micro-segment (uSID)
Size per ID	128 bits	16 bits
Encapsulation	One SID per list entry	Multiple IDs per 128-bit carrier

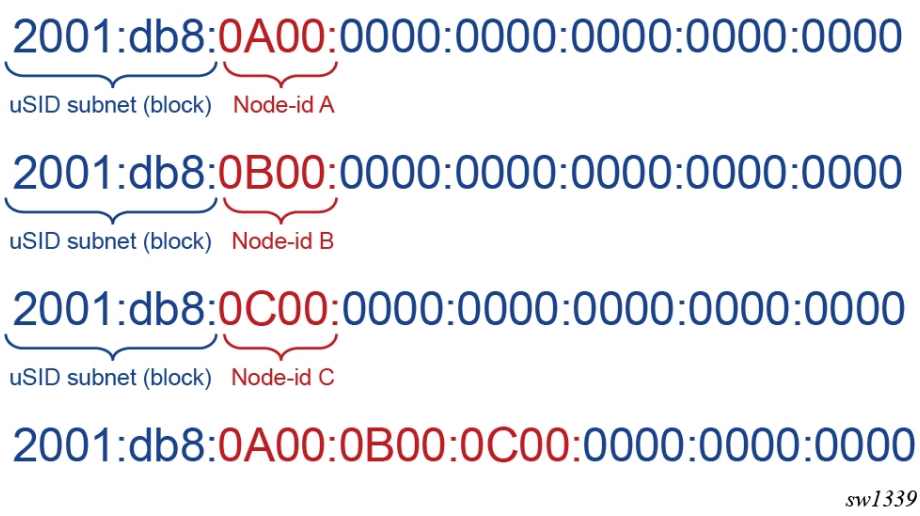
In regular SRv6, identifiers are assigned to nodes and form, together with the SID block, the LOCATOR part of any SID. SIDs are assigned per node and associated with a specific behavior (or function), resulting in the LOCATOR:FUNCTION structure.

Micro-segment SRv6 introduces a specific function (uN) which acts both as a locator (when combined with the block) and as a function (corresponding to END in regular SRv6), but without using any FUNCTION bits. In other words, the LOCATOR and END constructs of regular SRv6 correspond to a unique construct in micro-segment SRv6: <block><uN>.

Micro-segment SRv6 allows micro-SIDs to be compressed. Compression consists of coalescing several micro-SIDs into a 128-bit structure called a container.

The following figure shows three micro-SIDs, each identifying a different node in a network. The figure shows the result of compressing these three micro-SIDs in a container. Compression is possible because all micro-SIDs belong to the same block.

Figure 20: Compression of three micro-SIDs in a container



A container can be placed in the DA field of an IPv6 header and in a segment of the segment list of an SRH.

Compression allows the router to build longer paths with less overhead than with regular SRv6. However, it leads to specific datapath behaviors.

Micro-segment SRv6 introduces specific micro-segment SRv6 functions that correspond to functions defined for regular SRv6 (see Micro-segment SRv6 and BGP service control plane extensions).

In general, any SR Linux capability that applies to regular SRv6 also applies to micro-segment SRv6. In the documentation, only the differences are highlighted and commonalities are not repeated using micro-segment SRv6 specific terminology.

14.6 Configuring the SRv6 locator and SIDs

Procedure

An SRv6 SID is a 128-bit IPv6 address which follows the structure defined by RFC 8986:

SRv6 SID={LOCATOR:FUNCTION:ARGUMENT}

You must configure the main SRv6 subnet on the SRv6-enabled router.

The SRv6 locator is an IPv6 subnet (prefix and length) that provides reachability to all SIDs originated by this router using longest-prefix-match. The locator prefix, encoded in the LOCATOR field of the SRv6 SID, can be 64 or 96 bits long.The locator is subdivided into a SID block and a node ID.

For example, locator 3FFE:0:0:A1::/64 includes a SID block of 3FFE:0 and a node ID of 0:A1.

All nodes in an SRv6 domain must use locators and SIDs from the same SID block. In the previous example, the SID block is subnet 3FFE:0::/32.

SRv6 locators are created globally under **system.srv6.locator** and referenced per network-instance via **locator-name** in **network-instance.segment-routing.srv6.instance.locator** context.

One locator is required for algorithm 0 and for each IGP flexible algorithm (128-255). Multiple IGP instances can share the same locator for the same algorithm number.

The locator prefix (for example, 3FFE:0:0:A1::/64) is advertised in the SRv6 Locator TLV in IS-IS for both algorithm 0 and any configured flexible algorithm number, as defined in *draft-ietf-lsr-isis-srv6-extensions*.

It is also advertised as a prefix in the IP Reachability TLV (IS-IS TLV 236) for algorithm 0, allowing routers without SRv6 support to route packets to the next hop of the locator and ultimately to the destination node.

The FUNCTION field is user-configurable.

The ARGUMENT field is set to all zeros and cannot be configured. The ARGUMENT length must be signaled as zero in IS-IS End and End.X SIDs and in BGP service SIDs.

The combined length of LOCATOR and FUNCTION must not exceed 128. Within algorithm 0 and each IGP flexible algorithm, the locator function (FUNCTION field) assigns values to the End SID and End.X SID, which correspond to the node SID and the adjacency or adjacency SET SID.

The locator function also assigns values to the service SIDs owned by this node and advertised in the BGP control plane (End.DT4, End.DT6, End.DT46, and End.DX2).

The FUNCTION field can be divided into static and dynamic subranges. Use the static subrange to manually assign an SRv6 SID to a node, local adjacency, or service. IS-IS and BGP use the dynamic subrange to assign a SID to a local adjacency or service. The SID for an adjacency to an IS-IS neighbor over a broadcast interface (LAN End.X) is always dynamically assigned and cannot be configured.

Example: Configuring the SRv6 locator

In the following configuration example, the SRv6 locator `srl_loc1` is defined with a /48 prefix `2001:db8:100::/48` (block-length 48) and a 16-bit function field (function-length 16) using **algorithm 0**, allowing up to 1024 static function entries.

```
--{ + candidate shared default }--[  ]--
# info with-context system srv6 locator srl_loc1
    system {
        srv6 {
            default-source-address 2001:db8:100::1
            source-address 2001:db8:100::1 {
```

```
            }
            locator srl_loc1 {
                full-segment {
                    block-length 48
                    function-length 16
                    algorithm 0
                    prefix {
                        ip-prefix 2001:db8:100::/48
                    }
                    static-function {
                        max-entries 1024
                    }
                }
            }
        }
    }
```

Example: Configuring the SID

The SRv6 instances are configured per network-instance under **network-instance.segment-routing.srv6.instance**, with **id** in the range 1-2.

In the following configuration example, two static SIDs are provisioned in network instance : an End SID with function 100 (SRH mode psp) and an End-X SID with function 200 (SRH mode psp, protected, bound to sub-interface ethernet-1/1.0).

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default segment-routing srv6 instance 1
    network-instance default {
        segment-routing {
            srv6 {
                instance 1 {
                    locator srl_loc1 {
                        full-segment {
                            function {
                                end 100 {
                                    srh-mode psp
                                }
                                end-x 200 {
                                    srh-mode psp
                                    protection protected
                                    subinterface-name ethernet-1/1.0
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

14.7 Displaying regular SRv6 locator local SIDs

Procedure

You can use a show command to display information about regular SRv6 locator local SIDs.

Example

```
--{ + running }--[  ]--
# show system srv6 full-segment-locator local-sids
=============================================================================================

SRv6 End SIDS on Net-Inst: "default" SRv6 Inst: "1"

---------------------------------------------------------------------------------------------

+-------------------------------------------------------------------------+

| Locator Name                        Function     SID                     |

|                                     Value                                |

+=========================================================================+

| A1                                  201          2222:2:2:2:c9::         |

+-------------------------------------------------------------------------+

No. of end sids: 1

=============================================================================================

SRv6 Network Instance SIDS

S - Static Sids, D - Dynamic Sids

---------------------------------------------------------------------------------------------

+-------------------------------------------------------------------------------------------+

| Network Instance                    Inst    Locator Name                        Function
   Function     S     SID                        |

|                                                                                 Type
   Value        or                               |

|
                D                                |

+===========================================================================================+

| 1                                   1       A1                                  End-Dt46
   101          S     2222:2:2:2:65::            |

| 10                                  1       A1                                  End-Dt46
   110          S     2222:2:2:2:6e::            |

| 2                                   1       A1                                  End-Dt46
   102          S     2222:2:2:2:66::            |

| 3                                   1       A1                                  End-Dt46
   103          S     2222:2:2:2:67::            |

| 4                                   1       A1                                  End-Dt46
   104          S     2222:2:2:2:68::            |

| 5                                   1       A1                                  End-Dt46
   105          S     2222:2:2:2:69::            |
```

```
| 6                                   1       A1                                    End-Dt46
   106          S     2222:2:2:2:6a::           |

| 7                                   1       A1                                    End-Dt46
   107          S     2222:2:2:2:6b::           |

| 8                                   1       A1                                    End-Dt46
   108          S     2222:2:2:2:6c::           |

| 9                                   1       A1                                    End-Dt46
   109          S     2222:2:2:2:6d::           |

+-------------------------------------------------------------------------------------+

No. of network-instance sids: 10

=======================================================================================
Session has been idle, will logout in 300 seconds
--{ + running }--[  ]--
```

14.8 Configuring the micro-segment locator and SIDs

Procedure

SRv6 micro-segments are defined at the global level under **system.srv6.locator**, and each locator also has its own micro-segment definitions and SID functions under both **system.srv6.locator** and the per-instance hierarchy **network-instance.segment-routing.srv6.instance**.

The configuration of micro-SIDs follows the same logic as for regular SRv6 SIDs, except that:

- The user configures SIDs that are specific to micro-segment SRv6 (uA, uDT4, uDT6, uDT46).
- The configured value only needs to be unique on the system and not network wide.
- The resulting SID value is derived from the local range.
- Although the uN function is equivalent to the regular SRv6 END function, it is not configurable under the base instance because it is configured in the micro-segment-locator context.

Table 37: Global SRv6 micro-segment parameters

Parameters	Description
block-length	Operates the same as the standard SRv6 **block-length** version and the value must be the same on every platform network wide.
gib-size	Defines the maximum number of unique micro-segment locators that can be configured network wide. The value must be the same on every platform, network wide. This command splits in two the total number of values that can be encoded in 16 bits. A 16-bit field allows values in the range 0x0000 to 0xFFFF. Configuring 16 (default value) for the **global-sid-entries** value splits that range into the following:

Parameters	Description
	• 0x0000 to 0x3FFF for global identifiers • 0x4000 to 0xFFFF for local identifiers
sid-length	Defines the length of micro-SIDs. The only supported value is 16 (bits).

Table 38: Per-locator SRv6 micro-segment parameters (system level)

Parameters	Description
locator-name	SRv6 locator name
algorithm	IGP flexible algorithm ID
block	Reference to a pre-defined micro-SID block under `system srv6 micro-segment block`
srh-mode	Segment Routing Header (SRH) mode for uN
value	SRv6 uN function value.

Table 39: Per-locator SRv6 micro-segment (per network-instance / SRv6 instance)

Parameters	Description
id	SRv6 instance ID
locator-name	SRv6 locator name, referencing a system locator.
protection	Adjacency protection for automatic uA SID function
srh-mode	SRH mode for automatically allocated uA SIDs.
value	SRv6 SID function value for uA
value	SRv6 SID function value for uDT4
value	SRv6 SID function value for uDT6
value	SRv6 SID function value for uDT46
value	SRv6 SID function value for uDX2

Example: Configuring a global SRv6 micro-segment block

In the following configuration, a global SRv6 micro-segment block (`srl_blk`) is defined.

```
--{ + candidate shared default }--[  ]--
# info with-context system srv6 micro-segment
    system {
        srv6 {
            micro-segment {
                block-length 32
                gib-size 48
                sid-length 16
                block srl_blk {
```

```
                    prefix {
                        ip-prefix 2001:db9::/32
                    }
                    static-function {
                        max-entries 256
                    }
                }
            }
        }
    }
```

Example: Configuring an SRv6 locator and binding it to the micro-segment block

In the following configuration example, an SRv6 locator (`srl_loc_microsegment`) is defined and attached to the pre-defined micro-segment block (`srl_blk`).

```
--{ + candidate shared default }--[  ]--
# info with-context system srv6 locator srl_loc_microsegment
    system {
        srv6 {
            locator srl_loc_microsegment {
                micro-segment {
                    block srl_blk
                    algorithm 0
                    un {
                        srh-mode psp
                        value 1
                    }
                }
            }
        }
    }
```

Example: Configuring micro-segment SID functions

In the following configuration example, the locator (`srl_loc_microsegment`) is referenced and micro-segment SID functions are configured within a network instance SRv6 instance.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default segment-routing srv6 instance 1
    network-instance default {
        segment-routing {
            srv6 {
                instance 1 {
                    locator srl_loc_microsegment {
                        micro-segment {
                            function {
                                ua 10 {
                                    srh-mode psp
                                    protection protected
                                    subinterface-name ethernet-1/1.0
                                }
                                ua 100 {
                                    srh-mode psp
                                    protection protected
                                    subinterface-name ethernet-1/1.0
                                }
                            }
                        }
                    }
                }
            }
```

```
        }
    }
```

14.9 Displaying micro-segment SRv6 locator local SIDs

Procedure

You can use a show command to display information about micro-segment SRv6 locator local SIDs.

Example

```
--{ + running }--[  ]--
# show system srv6 micro-segment-locator local-sids
=============================================================================================

SRv6 un SIDS

---------------------------------------------------------------------------------------------

+-------------------------------------------------------------------------+

| Locator Name                       Function     SID                     |

|                                    Value                                |

+=========================================================================+

| A1                                 1            3336:1:1::              |

| A2                                 2            3336:2:2::              |

| A3                                 3            3336:3:3::              |

| A4                                 4            3336:4:4::              |

| A5                                 5            3336:5:5::              |

| A6                                 6            3336:6:6::              |

| C1micro                            3            1111:1:3::              |

+-------------------------------------------------------------------------+

No. of un sids: 7

=============================================================================================

SRv6 Network Instance SIDS

S - Static Sids, D - Dynamic Sids

---------------------------------------------------------------------------------------------

+-------------------------------------------------------------------------------------------
+

| Network Instance                   Inst    Locator Name                       Function
   Function     Oper         S     SID                       |
```

```
|                                                                                                  Type
   Value        Function     or                              |

|
                Value        D                               |

+=================================================================================================
+

| default                              1       A1                                      udt46
   301          16684        S     3336:1:1:412c::           |

| default                              1       A2                                      udt46
   302          16685        S     3336:2:2:412d::           |

| default                              1       A3                                      udt46
   303          16686        S     3336:3:3:412e::           |

| default                              1       A4                                      udt46
   304          16687        S     3336:4:4:412f::           |

| default                              1       A5                                      udt46
   305          16688        S     3336:5:5:4130::           |

| default                              1       A6                                      udt46
   306          16689        S     3336:6:6:4131::           |

| default                              1       C1micro                                 udt46
   101          16484        S     1111:1:3:4064::           |

+-------------------------------------------------------------------------------------------------
+

No. of network-instance sids: 7

=================================================================================================
```

14.10 Assigning loopback interface addresses from an SRv6 locator subnet

SR Linux supports assigning IPv6 addresses to a system (for example, system0) or a loopback interface (for example, lo0) drawn from a classic or micro-segment SRv6 locator prefix subnet.

To enable this feature, you must configure an IPv6 address for the system or loopback interface from a classic or a micro-segment locator by setting bits in the least significant octet of the locator, as described in Assigning an IPv6 address from a classic SRv6 locator and Assigning an IPv6 address from a micro-segment SRv6 locator. Only a /128 subnet mask is allowed for IPv6 addresses. Addresses from a flex-algo locator are not allowed.

A number of checks are performed in CPM to enforce the correct setting of the prefix bits and mask, and to verify the overlap with a locator subnet. See CPM support with classic SRv6 locator and CPM support with micro-segment SRv6 locator for more information.

A system or loopback interface configured to use an IPv6 address from an algorithm 0 locator address space behaves like any other system or loopback interface. It can be injected into IGP or BGP routing protocols, as well as into a VRF. Its address can be used as the source or destination address for control-plane protocol and OAM packets.

Note: This type of system or loopback interface cannot be added into non-default `network.instance`.

14.10.1 Assigning an IPv6 address from a classic SRv6 locator

When assigning an IPv6 address from a classic SRv6 locator, the feature reserves the least significant octet of the locator subnet for the IPv6 address of the system (for example, system0) and loopback interfaces (for example, lo0). This is the main subnet of the locator with bits to the right of the node ID set to zero, except for the interface bits.

The following is an example of the address assignment.

- locator subnet: fc01:1:0:105::/**64**
- locator subnet identifier: fc01:1:0:105:0:0:0:**0**
- system/loopback interface addresses allowed are: fc01:1:0:105:0:0:0:00[**01-FF**]/**128**

This feature supports only a/128 mask value. A maximum of 255 addresses can be allocated from the last octet of the locator. A value of zero for that octet represents the identifier of the entire subnet and is not allocatable.

14.10.1.1 CPM support with classic SRv6 locator

The following checks are performed in the CPM when assigning an IPv6 address from an SRv6 locator subnet. If any of the checks fail, the configuration fails.

- An address of the system interface or a loopback interface can only be drawn from the subnet of a classic SRv6 locator prefix assigned to algorithm 0.
- An address of the system interface or a loopback interface can only be drawn from a classic SRv6 locator prefix subnet that satisfies the following condition:

  {Prefix-length + function-length} < 120 bits
- An address of the system interface or a loopback interface must have all bits to the right of the classic SRv6 locator’s node ID field set to zero, except for the bits of the least significant octet.
- Creating a new locator or changing the prefix, function-length, or prefix-length of a configured locator is not allowed if an address from the locator prefix subnet is already assigned to the system or a loopback interface. Users must first delete the corresponding addresses from the system interface or loopback interfaces.

14.10.2 Assigning an IPv6 address from a micro-segment SRv6 locator

When assigning an IPv6 address from a micro-segment SRv6 locator, the feature reserves the least significant octet of the micro-segment locator subnet for the IPv6 address of system and loopback interfaces. This is the main subnet of the locator, with the bits to the right of the uN set to zero, except for the interface bits.

The following is an example of the address assignment.

- locator subnet: fc00:0:105::/**48**
- locator subnet identifier: fc00:0:105:0:0:0:0:**0**

- system/loopback interface addresses allowed are: fc00:0:105:0:0:0:0:00[**01-FF**]/**128**

This feature supports only a/128 mask value. A maximum of 255 addresses can be allocated from the last octet of the locator. A value of zero for that octet represents the identifier of the entire subnet and is not allocatable.

14.10.2.1 CPM support with micro-segment SRv6 locator

The following checks are performed in CPM when assigning an IPv6 address from a micro-segment SRv6 locator subnet. If any of the checks fail, the configuration fails.

- An address of the system interface or a loopback interface can only be drawn from the subnet of a micro-segment SRv6 locator prefix assigned to algorithm 0.
- An address of the system interface or a loopback interface can only be drawn from a micro-segment SRv6 locator prefix subnet that satisfies the following condition:

  {Block-length + sid-length(uN) + sid-length(uA or uDT)} < 120 bits
- An address of the system interface or a loopback interface must have all bits to the right of the micro-segment SRv6 locator's uN field set to zero, except for bits of the least significant octet.
- Creating a new locator, changing the uN value of a configured locator, or changing the block length of a configured locator are not allowed if an address from the locator prefix subnet is already assigned to the system or a loopback interface. You must first delete the corresponding addresses from the system interface or loopback interfaces.

14.10.3 Datapath support

The following procedures are supported in the datapath:

- When the user configures an IPv6 address for a system interface or a loopback interface, and that address is drawn from the subnet of a locally configured locator, the CPM adds the /128 prefix into the route table and into the FIB.
  - A packet matching the more specific route in the FIB for the interface is extracted to the CPM for local interface processing.
  - A packet matching the less specific locator route of the locator in the FIB undergoes SRv6 tunnel termination processing, as usual.
- Ingress PE and transit P routers forward packets destined for this system or the loopback interface using either the more-specific interface route, if advertised in IGP or BGP by the owner router, or the less-specific locator route from which the interface address is derived.
- A packet destined for the system or a loopback interface whose IPv6 address is drawn from a local locator prefix subnet can be received with an SRH in its encapsulation. An example use case of a packet received from a transit router that forwarded it over the remote LFA or TI-LFA repair tunnel of the corresponding less-specific locator route when the repair tunnel SIDs are in USP SRH mode.

Note: The datapath behavior for a micro-segment SRv6 locator is the same as that of a classic SRv6 locator.

14.11 IS-IS control plane extensions

The IS-IS control plane extensions for SRv6 are defined in *draft-ietf-lsr-isis-srv6-extensions*.

The IS-IS control plane advertises the SRv6 capabilities sub-TLV and the SRv6 Locator TLV. The latter includes the End function sub-TLV (equivalent to the prefix SID sub-TLV in SR-MPLS). IS-IS also advertises the End.X function sub-TLV (equivalent to the adjacency SID sub-TLV for a P2P link and a LAN in SR-MPLS) in the Extended IS Reachability TLV (top-level link TLV).

The weight fields in the End.X and LAN End.X sub-TLVs are not populated on transmit and are ignored on receipt of the link TLV.

The following table describes the supported IS-IS SRv6 TLVs in SR Linux.

Table 40: SRv6 IS-IS TLVs

SRv6 TLV/sub-TLV	Codepoint	IS-IS context TLV	Description	SR Linux support
SRv6 Capabilities sub-TLV	25	Router CAPABILITY TLV (242)	Indicates SRv6 support	Yes
SR-Algorithm sub-TLV	19	Router CAPABILITY TLV (242)	Indicates base algorithm 0 and Flex-Algo 128-255 support	Yes
Maximum Segments Left MSD Type sub-TLV	41	Router CAPABILITY TLV (242)	Indicates how deep a node terminating current segment can process Segments Left field of the SRH to move the next SID to outer IPv6 header DA field	Yes Advertised value = 7 SIDs Received TLV is displayed in the LSDB but is not used for any purpose
Maximum End Pop MSD Type sub-TLV	42	Router CAPABILITY TLV (242)	Maximum number of SIDs in a SRH when a node removes the SRH (Penultimate Segment Pop (PSP) or Ultimate Segment Pop (USP) modes of SRH)	Yes Advertised value = 7 SIDs Received TLV is displayed in the LSDB but is not used for any purpose
Maximum H.Encaps MSD type sub-TLV	44	Router CAPABILITY TLV (242)	Indicates the maximum number of SIDs in an SRH that a router can push when forwarding an IP or L2 packet over a SRv6 policy	Yes Advertised value = 7 SIDs Received TLV is displayed in the LSDB but is not used for any purpose

SRv6 TLV/sub-TLV	Codepoint	IS-IS context TLV	Description	SR Linux support
Maximum End D MSD Type Sub-TLV	45	Router CAPABILITY TLV (242)	The maximum number of SIDs in an SRH when a node removes the SRH and performs the End.DX2/4/6 or End.DT4/6 function (USP mode of SRH)	Yes Advertised value = 7 SIDs Received TLV is displayed in the LSDB but is not used for any purpose
SRv6 Locator TLV	27	Is a Top-level IS-IS TLV	Advertises the locator prefix configured on this node to terminate SIDs in algorithm 0 and flex-algo 128-255	Yes
SRv6 End SID sub-TLV	5	SRv6 Locator TLV	Advertises the SID for the endpoint or End function (equivalent to the prefix SID sub-TLV in SR-MPLS)	Yes
Prefix Attribute Flags Sub-TLV	4	SRv6 Locator TLV (Also in IP Reach TLV 236)	Provides attributes of a prefix leaked between IS-IS levels	Yes
SRv6 End.X SID sub-TLV	43	Top-level Extended IS reachability TLV (22)	Advertises the SID for the adjacency over a P2P link (equivalent to the adjacency SID sub-TLV for P2P link in SR-MPLS)	Yes
SRv6 LAN End.X SID sub-TLV	44	Top-level Extended IS reachability TLV (22)	Advertises the SID for the adjacency over a LAN (equivalent to the adjacency SID sub-TLV for LAN in SR-MPLS)	Yes
SRv6 SID Structure Sub-Sub-TLV	1	SRv6 End SID Sub-TLV SRv6 End.X SID Sub-TLV SRv6 LAN End.X SID Sub-TLV	Provides the length of each field (Block, Locator, Function, and Argument) of the SRv6 SID that it is advertised with	No SR Linux does not advertise this sub-sub-TLV. If received from other vendor's implementation, it is not displayed in the Link-State database and is

SRv6 TLV/sub-TLV	Codepoint	IS-IS context TLV	Description	SR Linux support
				also not propagated with the locator TLV

When both SR-MPLS and SRv6 are enabled on the same IS-IS instance, an MPLS node SID cannot be configured for a prefix of the locator or an End SID. This is because the SRv6 locator subnet cannot be added to a network interface and an MPLS node SID is configurable against a network interface only.

However, both the SRv6 locator tunnel and the SR-MPLS tunnel are programmed if IS-IS receives a /128 prefix that has both a locator TLV and a prefix SID TLV (with the node flag enabled) from a third-party router implementation. If the prefix SID is for a subnet larger than /128, only the SRv6 locator tunnel is programmed and the SR-MPLS tunnel is not.

Each function encoded in an SRv6 SID has its own endpoint behavior codepoint as listed in Table 41: SRv6 SID function endpoint behavior codepoints.

Note: SRv6 standards provide the flexibility to advertise transport and service SIDs in both IS-IS and BGP. In SR Linux, the IS-IS control plane only advertises the transport SID functions described in Table 41: SRv6 SID function endpoint behavior codepoints and only uses the transport SIDs advertised in IS-IS for building LFA repair tunnels.
SR Linux advertises service SID functions in the BGP control plane only as described in BGP service control plane extensions.

For the SRH processing and removal at the SID termination, the following modes of operation are associated with the termination of the End or End.X SID. These modes are sometimes referred to as SID flavors and IS-IS assigns a unique codepoint for each mode of the End or End.X SID of a given adjacency.

- **Basic or unflavored SRH mode**

  The router that terminates an End or End.X SID and the Segments-Left field in the received packet is 0 before decrementing, keeps the SRH in the packet, and processes the packet identified by the next-header in the SRH. Typically, the next-header indicates another SRH and the packet is then forwarded based on the lookup of that next-SID; SR Linux does not support this mode.

- **Ultimate Segment Pop (USP) SRH mode**

  The egress PE terminates the last segment in the outer IPv6 header, removes the SRH and processes the inner service or control plane packet as indicated by the SRH next-header field. SR Linux supports this mode.

- **Penultimate Segment Pop (PSP) SRH mode**

  The router that terminates the End or End.X segment before the last in the segment list, meaning the Segments-Left field before decrementing has a value of 1, removes the SRH on behalf of the egress PE. SR Linux supports this mode.

- **Ultimate Segment Decapsulation (USD) mode (psp-usd, usp-usd, and psp-usp-usd flavors)**

  This is a variant of the USP mode in which the node removes the outer IPv6 header and associated SRH and moves directly to process the inner IPv6 packet as indicated by the SRH next-header field. This mode is used to terminate a TI-LFA or a Remote LFA repair tunnel originated with the H.Encaps.Red encapsulation.

  An SRv6 ingress PE or transit P router normally uses the H.Insert.Red behavior to build an LFA (remote LFA or a TI-LFA) repair tunnel. This inserts a dedicated SRH to carry the additional node and

adjacency SIDs of the repair tunnel. SR Linux supports originating and terminating repair tunnels using H.Insert.Red behavior.

Some third-party implementations use the H.Encaps.Red behavior, which inserts a dedicated outer IPv6 header and SRH to carry the additional SIDs of the LFA repair tunnel. While SR Linux does not support originating an LFA repair tunnel with the H.Encaps.Red behavior, it does support terminating the repair tunnel.

To allow third-party implementations to activate the H.Encaps.Red repair tunnel, IS-IS in SR Linux can signal support for the USD mode of the outer IPv6 header and SRH processing with End and End.X SIDs in classic SRv6, and with uN and uA in micro-segment SRv6.

You can configure the advertisement of the **psp-usd**, **usp-usd**, or **psp-usp-usd** flavor of the USD mode for each configured and auto-allocated node SID or adjacency SID. The **psp-usp-usd** behavior is similar to the **psp-usd** behavior because the datapath in SR Linux always performs the USP SRH mode when Segments-Left = 0 on a received SRv6 packet.

Table 41: SRv6 SID function endpoint behavior codepoints

SID function endpoint behavior	Codepoint (per RFC 8986)	SID type: End.SID	SID type: End.X SID	SID type: LAN End.X SID	Advertising protocol	Supported
End (PSP, USP, USD)	1-4, 28-31	Yes	No	No	IS-IS	Yes IS-IS only (PSP value = 2, USP value = 3) (PSP&USD value = 29, USP&USD value = 30) (PSP&USP&USD value = 31)
End.X (PSP, USP, USD)	5-8, 32-35	No	Yes	Yes	IS-IS	Yes IS-IS only (PSP value = 6, USP value = 7) (PSP&USD value = 33, USP&USD value = 34) (PSP&USP&USD value = 35)
End.T	9-12, 36-39	Yes	No	No	IS-IS	No[3]

[3] IS-IS saves SID sub-TLVs for endpoint behavior values that it does not support when received from a third-party implementation. However, it only uses End and End.X endpoint behaviors in RLFA and TI-LFA. BGP advertises the supported endpoint behaviors End.DT4, End.DT6, End.DT46, and End.DX2 and accepts any behavior codepoint with a supported NLRI type.

SID function endpoint behavior	Codepoint (per RFC 8986)	SID type: End.SID	SID type: End.X SID	SID type: LAN End.X SID	Advertising protocol	Supported
(PSP, USP, USD)						
End.DX6	16	No	Yes	Yes	BGP or IS-IS	No[3]
End.DX4	17	No	Yes	Yes	BGP or IS-IS	No[3]
End.DT6	18	Yes	No	No	BGP or static	Yes[3] BGP only
End.DT4	19	Yes	No	No	BGP or static	Yes[3] BGP only
End.DT46	20	Yes	No	No	BGP or static	Yes[3] BGP only
End.DX2	21	Yes	No	No	BGP or static	Yes[3] BGP only

14.11.1 Micro-segment SRv6

The following table lists the TLVs that micro-segment SRv6 supports in addition to the TLVs described in Table 40: SRv6 IS-IS TLVs.

Table 42: SRv6 IS-IS TLVs additionally supported by micro-segment SRv6

SRv6 TLV/sub-TLV	Codepoint	IS-IS context TLV	Description	SR Linux support
SRv6 SID Structure sub-sub-TLV	1	SRv6 uN SID Sub-TLV, SRv6 uA SID Sub-TLV, SRv6 LAN uA SID Sub-TLV	Provides the length of each field (block, locator, function, and argument) of the SRv6 uSID it is advertised with	Yes

The following table lists the endpoint behavior codepoint for each SID function encoded in a micro-segment SRv6 SID.

Table 43: SRv6 SID function endpoint behavior codepoints

SID function endpoint behavior	Codepoint	SID type: End.SID	SID type: End.X SID	SID type: LAN End.X SID	Advertising protocol	Supported
uN	42-50	Yes	No	No	IS-IS	Yes [4] IS-IS only (PSP value = 44, USP value = 45) (PSP&USD value = 48, USP&USD value = 49) (PSP&USP&USD value = 50)
uA	51-59	No	Yes	Yes	IS-IS	Yes [4] IS-IS only (PSP value = 53, USP value = 54) (PSP&USD value = 57, USP&USD value = 58) (PSP&USP&USD value = 59)

14.11.2 Enabling micro-segment SRv6 in the IS-IS instance

Procedure

To configure micro-segment SRv6 in the IS-IS instance, use the command, **network-instance protocols isis instance segment-routing srv6**.

Example

In the following configuration example, the IS-IS instance (**instance** 1) has micro-segment SRv6 enabled and references the SRv6 locator **srl_loc_microsegment** for Level 1 routing, advertising it with a metric of 5 and a route tag of 10.

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance 1
```

[4] IS-IS saves SID sub-TLVs for endpoint behavior values that it does not support when received from a third-party implementation. However, it only uses End and End.X endpoint behaviors in RLFA and TI-LFA.

BGP advertises the supported endpoint behaviors End.DT4, End.DT6, End.DT46, and End.DX2 and accepts any behavior codepoint with a supported NLRI type.

```
    network-instance default {
        protocols {
            isis {
                instance 1 {
                    segment-routing {
                        srv6 {
                            locator srl_loc_microsegment {
                                level-capability L1
                                tag 10
                                level 1 {
                                    metric 5
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

14.12 SRv6 support in IS-IS Multi-Topology (MT)

SR Linux supports SRv6 for Multi-Topology IPv6 (MT2) in IS-IS.

You must first configure a locator for use in IS-IS IPv6 unicast multi-topology. You can enable one or more SRv6 local locators in a specific IS-IS instance. Each locator can be enabled in a single topology of an IS-IS instance, topology 0 (MT0) or topology 2 (MT2). By default, a locator name added to an IS-IS instance is enabled in MT0. A local locator can be used in multiple IS-IS instances and can be assigned to at most one IPv6 topology independently within each IS-IS instance.

MT-ID 0 supports the standard IPv4 and IPv6 topologies, whereas MT-ID 2 supports only IPv6.

14.12.1 Configuring IS-IS Multi-Topology (MT) for SRv6

Procedure

Use the command, **network-instance protocols isis instance segment-routing srv6 locator multi-topology** to configure a locator.

Example

In the following configuration example, the SRv6 locator (`srl_loc1`) added to an instance is enabled for **multi-topology-2**.

Note:
Unlike MT0, which is the default for IPv4, MT2 is not enabled by default and must be configured to manage IPv6 traffic. You must explicitly enable IPv6 routing in an IS-IS instance (**network-instance protocols isis instance ipv6-unicast admin-state enable**)

```
--{ + running }--[  ]--
# info with-context network-instance default protocols isis instance 1 segment-routing
 srv6 locator srl_loc1 multi-topology
    network-instance default {
        protocols {
            isis {
```

```
                instance 1 {
                    segment-routing {
                        srv6 {
                            locator srl_loc1 {
                                multi-topology {
                                    multi-topology-0 false
                                    multi-topology-2 true
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

14.12.2 IS-IS control plane changes

When a local locator is enabled in the MT2 IPv6 unicast topology of an IS-IS instance, IS-IS advertises the following routes:

- The prefix in a top-level Multi-topology Reachable IPv6 Prefixes TLV type 237 with the MT-ID field set to 2.
- A top-level SRv6 Locator TLV type 27 that contains the locator prefix as well as the End SID sub-TLVs associated with this local locator. The locator TLV is advertised with the MT-ID field set to 2.
- The End.X or LAN End.X sub-TLV in the top-level Multi-topology Extended IS Reachability TLV (222).
- The TE Application Specific Link Attributes sub-TLV in the top level Multi-topology Extended IS Reachability TLV (222), which supports the flex-algo feature.
- The Application-Specific Shared Risk Link Group (SRLG) TLV (238).

The following table summarizes the new or modified TLVs in support of SRv6 in IS-IS MT2.

Table 44: SRv6 IS-IS MT2 TLVs

Multi-topology TLV	Codepoint	IS-IS context TLV	Description	SR Linux support
Multi-topology IPv6 Reach TLV	237	Is a top-level IS-IS TLV	The main prefix TLV MT-ID field set to 2	Yes
SRv6 Locator TLV	27	Is a top-level IS-IS TLV	Indicates that the locator, configured on this node, is used to terminate SIDs in algorithm 0 and flex-algo 128-255 MT-ID field set to 2	Yes
SRv6 End SID Sub-TLV	5	SRv6 Locator TLV	Advertises the SID for the endpoint or End function (equivalent to the prefix SID in SR-MPLS)	Yes
Prefix Attribute Flags Sub-TLV	4	SRv6 Locator TLV	Indicates that the prefix of the locator is anycast	Yes

Multi-topology TLV	Codepoint	IS-IS context TLV	Description	SR Linux support
		(also in IPv6 Reach TLV 237)		
SRv6 End.X SID Sub-TLV	43	Top-level Multi-topology Extended IS Reachability TLV (222)	Advertises the SID for the adjacency or End.X function over a P2P link (equivalent to the adjacency SID sub-TLV for P2P link in SR-MPLS)	Yes
SRv6 LAN End.X SID sub-TLV	44	Top-level Multi-topology Extended IS Reachability TLV (222)	Advertises the SID for the adjacency or End.X function over a LAN (equivalent to the adjacency SID sub-TLV for LAN in SR-MPLS)	Yes
SRv6 SID Structure Sub-Sub-TLV	1	SRv6 End SID Sub-TLV, SRv6 End.X SID Sub-TLV, SRv6 LAN End.X SID Sub-TLV	Provides the length of each field (Block, Locator, Function, and Argument) of the SRv6 SID that it is advertised with	No SR Linux does not advertise this sub-sub-TLV. If received from other vendor's implementation, it is not displayed in Link-State database and is also not propagated with the locator TLV.
Application Specific Link Attributes (ASLA) sub-TLV	16	Top-level Multi-topology Extended IS reachability TLV (222)	Advertises the link attributes for the flex-algo application	Yes
Application-Specific Shared Risk Link Group (SRLG) TLV	238	Is a top-level IS-IS TLV	Advertises the link SRLG attribute for the flex-algo application	No The SRLG constraint is not supported in IS-IS MT0 or MT2. IS-IS does not advertise this TLV and does not use it for any purpose if received from the network.

In prior releases of SR Linux, only local locator routes and local prefix routes were redistributed with an export policy between topologies MT0 and MT2 of the same IS-IS instance or different IS-IS instances. These local locators were redistributed as prefix TLVs and not as locator TLVs. In addition, the tag of the local locator was not redistributed. Routes of remote locators and remote prefixes were not redistributed.

To extend SRv6 at a boundary of two IS-IS instances operating IPv6 in different topologies, the SRv6 support in the MT2 feature enhances this behavior as follows:

- It allows the concurrent enabling of a local locator in multiple IS-IS instances and in either MT0 or MT2 topology of each of these IS-IS instances.
- It allows the redistribution with an export policy of remote locator routes between MT0 and MT2 topologies of different IS-IS instances. The locator routes are redistributed as locator TLVs. In algorithm 0, a prefix TLV is advertised in addition to the locator TLV.

Redistribution between MT0 and MT2 topologies of the same IS-IS instance is not allowed for either remote locator or remote prefix routes.

14.12.3 Locator and SID resolution

The MT2 local locator, remote locator, SID resolution, and programming into the tunnel table and datapath tunnel follow the same rules as those for enabling SRv6 in the single topology (MT0).

If a remote IPv6 prefix route is received in both MT0 and MT2, the route with the lowest cost (IGP metric) is selected. If both routes have the same metric, the MT0 route is selected. The selected route is programmed in the route table and FIB. If that prefix also advertised a locator TLV, the corresponding SRv6 route is updated in the route table and FIB to point to the SRv6 tunnel, which is programmed in the tunnel table.

Note:
The preference of the MT0 route over the MT2 route is solely based on comparing the cost of each route. So, a remote IPv6 prefix route without a locator TLV can win over a remote IPv6 route with a locator TLV. The programmed route in the route table is a regular IS-IS route and does not have an SRv6 tunnel associated with it.

The same selection rule applies to a locator TLV advertised with a flex-algo number in both MT0 and MT2. The selection is based on comparing the cost of the routes using the metric of that specific algorithm (IGP, TE, or latency metric). The selected SRv6 route is added to the tunnel table.

14.13 SRv6 locator summarization with IS-IS

If an IS-IS domain exists out of multiple areas, you must redistribute SRv6 locators between areas for inter-area SRv6-based transport services. Each SRv6 locator is associated with an applied topology algorithm as follows:

- algorithm 0 for the base SPF topology
- algorithm in the range of 128 to 255 for flexible algorithms

Scaling is impacted when all existing SRv6 locators are redistributed between all existing areas. SRv6 locator summarization with IS-IS reduces the sizes of the LSDB and the IPv6 routing table and increases network stability.SRv6 locators can be summarized when they are redistributed from one area to another. This helps to reduce the number of SRv6 locators existing in each area.

The following apply when the summary address is configured and the route table has matching member prefixes:

- For algorithm 0, when algorithm 0 is not explicitly configured, the summary is inserted as an IS-IS IP Reachability TLV and all more specific prefixes from both the IP Reachability TLV and SRv6 locator TLV are suppressed.
- For algorithm 0, when algorithm 0 is explicitly configured, the summary is inserted as an IS-IS IP Reachability TLV and as an SRv6 locator TLV, and all more specific prefixes from both the IP Reachability TLV and the SRv6 locator TLV are suppressed.
- For algorithms in the range of 128 to 255, the summary is inserted as an SRv6 locator TLV and all more specific locator prefixes in the SRv6 locator TLV are suppressed.
- The summary address SRv6 locator does not contain any END SIDs of member SRv6 locator prefixes in the SRv6 locator TLV.

The following apply when using administrative tags for SRv6 locator summaries:

- When SRv6 locators are summarized from one IS-IS level into another IS-IS level, special care must be taken to avoid re-redistributing them back into the original IS-IS level and potentially causing routing loops. Routing filters must be used to prevent such routing loops.
- Existing filters can use 32-bit administrative tags to match upon routes and avoid routing loops. This route tag can be set using the **network-instance protocols isis instance inter-level-propagation-policies level1-to-level2 summary-address** command when originating an algorithm-aware SRv6 summary locator.
- A routing policy with a match tag supports matching to both classic IPv6 prefix tags and SRv6 locator tags.

14.13.1 Configuring an algorithm-aware SRv6 locator with summarization

Procedure

A summary SRv6 algorithm-aware locator address is configured using the **network-instance protocols isis instance inter-level-propagation-policies level1-to-level2 summary-address** command.

Example

In the following configuration example, the **level1-to-level2** policy instructs IS-IS to aggregate all more-specific IPv6 prefixes within 2001:db8:100::/48 into a single prefix when propagating routes from Level 1 to Level 2.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default protocols isis instance 1 inter-level-
propagation-policies level1-to-level2
    network-instance default {
        protocols {
            isis {
                instance 1 {
                    inter-level-propagation-policies {
                        level1-to-level2 {
                            summary-address 2001:db8:100::/48 {
                            }
                        }
                    }
                }
            }
```

```
        }
    }
```

14.14 IS-IS Flex-Algorithm for SRv6

SRv6 introduces flexible algorithms to the IPv6 data plane.

A router is provisioned with topology- or algorithm-specific locators for each topology or algorithm pair supported by that node. Each locator is a covering prefix for all SIDs provisioned on that router that have the matching topology or algorithm. Locators associated with flexible algorithms are not advertised in a Prefix Reachability TLV (236 or 237). However, locators associated with algorithm 0 are advertised in a Prefix Reachability TLV (236 or 237), which allows legacy routers that do not support SRv6 to install a forwarding entry for algorithm 0 SRv6 traffic.

Each SRv6 locator is associated with an algorithm (either algorithm 0 or a flexible algorithm in the range of 128 to 255) and each algorithm represents a topologically-constrained forwarding construct. The M-flag within the flexible algorithm prefix metric sub-TLV is not applicable to prefixes advertised as SRv6 locators. The metric field in the locator TLV is used regardless of the M-flag in the FAD advertisement. A router configured to participate in a flexible algorithm must use the selected FAD to compute the corresponding routing table. The available options are as follows:

- Algorithm 0 (legacy routing table entries) is constructed from information advertised as a traditional IP Reachability TLV or as an SRv6 locator TLV (27). When IP reaches TLV and the SRv6 locator TLV contains conflicting information, then the IP Reachability TLV information is used.
- Algorithms ranging from 128 to 255 (Flex-Algorithm routing table entries) are constructed from information advertised and constructed from locators found in the SRv6 locator TLV (27).

For route leaking of flexible algorithm-aware SRv6 locators between IS-IS areas, the following rules apply when a topology TLV (IP Reachability TLV or SRv6 locator TLV) is leaked, including leaked locators and end SIDs:

- For algorithm 0, this SRv6 locator route is programmed as a regular IS-IS route. If an IS-IS route is readvertised and also has an SRv6 locator TLV, it is readvertised as a regular IP Reachability TLV and SRv6 locator TLV.
- For algorithms ranging from 128 to 255, if locator leaking is enabled, the original SRv6 locator TLV is readvertised as an SRv6 locator TLV into the other area.
- The default locator leaking behavior between levels is as follows:
- – For Level 1 to Level 2, leaking is enabled by default.
  – For Level 2 to Level 1, leaking is disabled by default.

If a locator is associated with a flexible algorithm and the LFA is enabled, then LFA paths to the locator prefix must be calculated using the flexible algorithm in the corresponding topology to guarantee that they follow the same constraints as the calculation of the primary paths. LFA paths must only use SRv6 SIDs advertised specifically for the flexible algorithm. The LFA configuration is inherited from algorithm 0. The anycast behavior of SRv6 flexible algorithms is inherited from the standard algorithm 0 (standard SPF) SRv6 configuration.

The IS-IS neighbor advertisements are topology-specific and not algorithm-specific. Therefore, the SRv6 End.X SIDs inherit topology from the associated neighbor advertisement, but the algorithm is specified in the individual SID. All End.X SIDs are a subnet of a locator with matching topology and algorithm, which is advertised by the same node in an SRv6 locator TLV. The End.X SIDs that do not meet this requirement

are ignored. All End.X SIDs must find a supernet by the subnet of a locator with the matching algorithm, which is advertised by the same router in an SRv6 locator TLV. The End.X SIDs that do not meet this requirement are ignored.

IS-IS protocol limitations affect enabling SRv6 flexible algorithms on a broadcast network. On a broadcast network, the LAN End.X SIDs of all neighbors for all participating flexible algorithms need to be advertised in a single LSP fragment because each IS-IS TE-NBR with all its TLV blocks must be advertised in one IS-IS LSP fragment. The amount of information inserted by segment routing for SRv6 into the LSP fragment depends upon the number of flexible algorithms used, the number of static or auto-end.X configured per locator, and if both SRv6 and SR-MPLS are deployed.

14.15 Loop-Free Alternate (LFA) support

LFA, remote LFA, and Topology-independent LFA (TI-LFA) are supported in the following router roles:

- service originating role
- transit role with segment termination
- transit role without segment termination

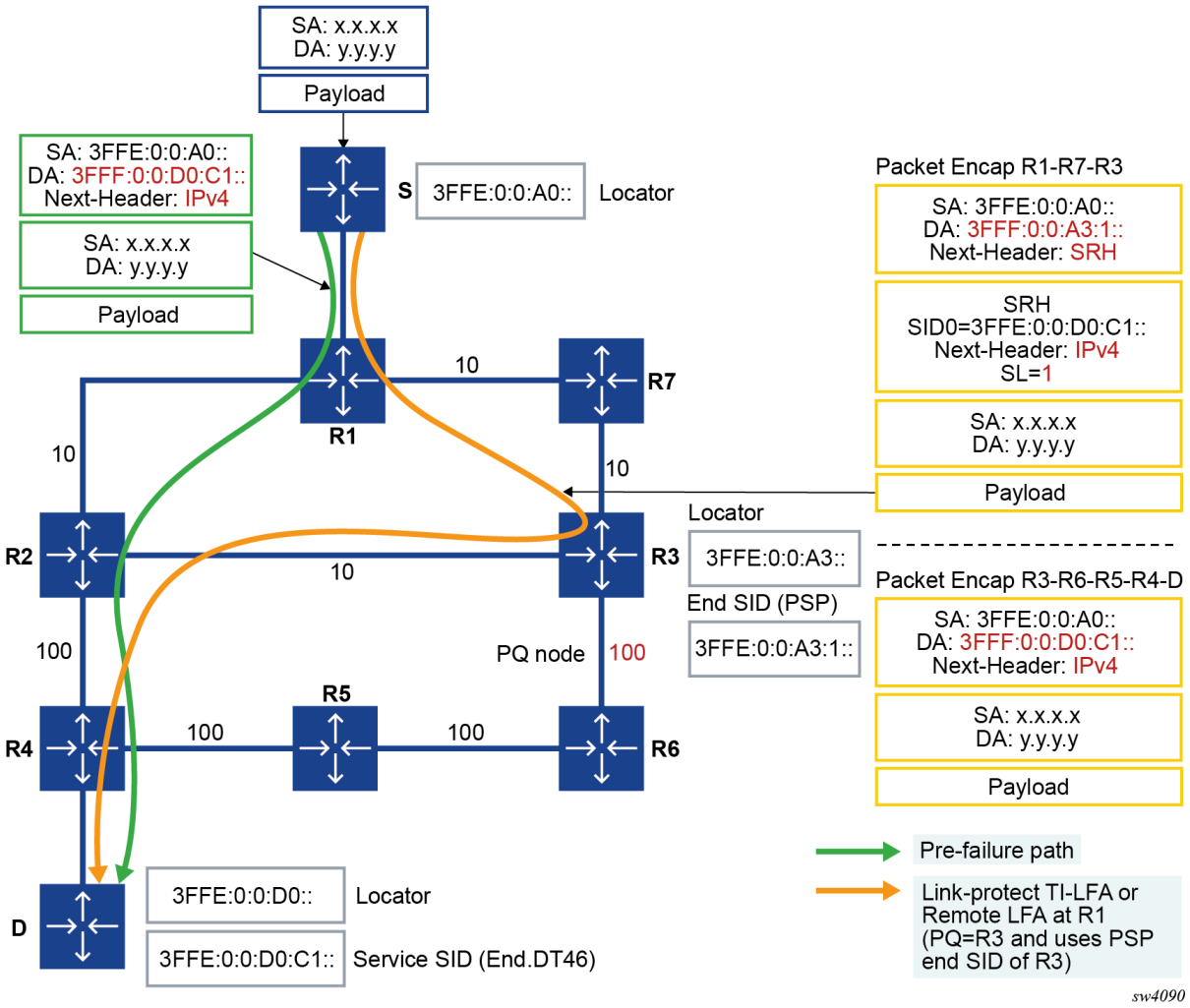
The backup is computed and programmed for each remote SRv6 node SID, service SID (for example, End.DT4, End.DT6, End.DT46, and End.DX2), and for each local adjacency or LAN adjacency SID.

A base LFA backup path or a TI-LFA backup path that uses a direct IP next hop (not a repair tunnel), requires configuring a next hop that is different from the primary path and does not modify the SID list pushed on the primary path.

When the RLFA or TI-LFA backup path uses a repair tunnel (source routed or not), the additional SIDs of the repair tunnel must be inserted into the packet when the backup is activated. This requires the insertion of an LFA dedicated SRH into the packet. The SRv6 behavior is referred to as H.Insert.Red and is described in *draft-filsfils-spring-srv6-net-pgm-insertion-04*. The application of this behavior to the LFA repair tunnel is described in *draft-voyer-6man-extension-header-insertion-10*.

The following figure shows the packet encoding for the primary and remote LFA backup path using the H.Insert.Red behavior and pushing a dedicated reduced SRH for the repair tunnel. The figure uses regular SIDs but is equally applicable to micro SIDs.

Figure 21: LFA repair tunnel packet encoding



The base LFA, remote LFA, and TI-LFA features operate with SRv6 tunnels in the same way as with SR-MPLS tunnels.

To enable LFA on an IS-IS instance, use the **loopfree-alternate admin-state enable** command. For details, see Configuring LFA.

Remote LFA (RLFA) and TI-LFA are, configured with **remote-lfa** and **ti-lfa** options under **network-instance default protocols isis loopfree-alternate** context. For details, see Configuring Remote LFA and Configuring TI-LFA.

Note:
The TI-LFA algorithm prefers a micro-segment locator over a regular locator when both are present. As such, a micro-segment path may protect a regular SRv6 path.

14.16 Route table, FIB table, and tunnel table support

The following tables and information are needed to process a SRv6 packet at service origination, service termination, and transit router roles.

Route table and FIB

SRv6 locator and SID resolution is performed in the RTM and forwarding of all SRv6 packets is performed in the FIB.

The TTM is used to save details of the SRv6 tunnel but is not used directly to forward user or CPM originated packets.

The RTM and FIB are programmed with the routes of the local and remote locators, the local End.X SIDs, and the local End SIDs.

When a policy is applied to export SRv6 routes from the route table to another IS-IS instance, only the IP Reachability TLV and the locator TLV, along with the End SID sub-TLVs, are advertised by the receiving IS-IS instance. Local End, End.X, and LAN End.X routes are not exported nor advertised as separate routes.

- **remote locator (route owner = SRv6 IS-IS)**

  All routers in the SRv6 domain populate a resolved remote locator prefix received in the SRv6 Locator TLV in the route table and the FIB.

  A SRv6 packet is always forwarded out in the datapath using the FIB.

  For algorithm 0, the same prefix is advertised with the IP reach prefix TLV and the SRv6 Locator TLV. A single route entry is programmed in the route table and the FIB.

  The prefix of an IGP flexible algorithm locator TLV is never advertised with an IP reach prefix TLV. Therefore, the route of the locator TLV is programmed in the route table and the FIB.

  – **remote locator with up to 64 ECMP next hops**

    IS-IS models a remote locator prefix with two or more ECMP next hops as an IGP route with tunneled next hops using a protected NHLFE with hardware PG-ID per tunneled next hop.

    This implementation provides uniform failover in ECMP. IS-IS allocates a hardware PG-ID to each next hop it establishes an adjacency with. That PG-ID is then used when programming SRv6 routes of a remote locator and of a local adjacency that resolve to this next hop.

    The route table manager programs the route into the FIB. IS-IS creates an SRv6 tunnel for the locator prefix. The tunnel is added to the tunnel table. The IS-IS route entries in the route table and the FIB point to the tunnel ID of this tunnel in the tunnel table.

    Weighted ECMP, when enabled on the interfaces of this IS-IS instance, is supported when forwarding packets over the locator next hops.

  – **remote locator with primary or backup next hops**

    To provide uniform failover, IS-IS models a remote locator prefix with a primary next hop or a primary and LFA backup next-hop pair as an IGP route with a tunneled next hop using a protected NHLFE with a hardware PG-ID.

    The route table manager programs the route into the FIB. IS-IS creates a SRv6 tunnel for the locator prefix. The tunnel is added to the tunnel table. The IS-IS route entries in the route table and the FIB point to the tunnel ID of this tunnel in tunnel table.

- **local locator (route owner = SRv6)**

All routers in the SRv6 domain populate a route entry in the route table and the FIB to terminate packets destined for the local locator. This is modeled like any other local route but with the SRv6-specific route owner.

- **local adjacency SID (route owner = IS-IS)**

  All routers in the SRv6 domain populate a route entry in the route table and FIB for each local End.X and LAN End.X adjacency SID with primary and backup next hops.

  The route table and FIB entries are modeled like a remote locator prefix with primary and backup next hops.

- **local End SID (route owner = SRv6)**

  All routers in the SRv6 domain populate a route entry in the route table and the FIB to terminate packets destined for each local End SID. This is modeled like any other local route, but with the SRv6-specific route owner.

With micro-segment SRv6, entries pertaining to local services populate the RTM and FIB. In the FIB, those entries are aggregated. At most, 15 entries are needed to cover the whole service addressing space offered by micro-segment SRv6. The route owner is SRv6.

14.16.1 Tunnel Table Manager (TTM)

The tunnel table is not used directly in the SRv6 locator resolution, in SID resolution, or in packet forwarding. All resolution is performed in the RTM and forwarding is performed in the FIB.

Each resolved remote locator or local adjacency creates an entry in TTM (ROUTE_OWNER_SRV6_ISIS). The entries for remote locators and local adjacencies in the RTM and FIB point to the tunnel ID of this tunnel. The TTM entry is not used directly for forwarding SRv6 packets. However, the route resolution behavior for SRv6 services can be modified to preferentially resolve in TTMv6.

14.16.2 Users of route table SRv6 routes

The SRv6 locator and adjacency routes in the route table can be used to forward the following user and CPM originated packets:

- user packets
- ICMPv6 echo request and echo reply packets as described in the SR Linux OAM and Diagnostics Guide
- UDP traceroute packets as described in the SR Linux OAM and Diagnostics Guide

Forwarding and terminating of any other CPM originated packets are not supported. Specifically, received management protocol and control plane protocol packets that are encapsulated in SRv6 are dropped.

If you configure the address of a BGP neighbor, an LDP peer, or an RSVP-TE LSP destination to match a locator prefix or a SID, packets are forwarded over the SRv6 tunnel but are dropped at the destination router.

14.17 Datapath support

This section describes packet processing in the datapath on ingress PE, egress PE, and transit P router roles.

SR Linux supports both regular SRv6 and micro-segment SRv6. Both modes can operate concurrently on the same platform, but it requires that the SIDs of each type come from different SID blocks.

14.17.1 Service origination and termination roles

The SRv6 processing is performed inline in the IP datapath when forwarding service packets over a shortest path SRv6 tunnel at service origination or when terminating an SRv6 packet with the service SID in the DA field of the outer IPv6 header.

14.17.1.1 At the ingress PE

The following occurs at the ingress PE:

- **Packet forwarded to a shortest path SRv6 tunnel**

  The datapath pushes the SRv6 encapsulation header on the received Layer 2 or Layer 3 service packet.

  – The Outer DA field is set to service SID (End.DT4, End.DT6, End.DT46, or End.DX2 SID).

  – The Outer next-header field is set to IPv4, IPv6, or Ethernet.

  – The hop-limit field in the outer IPv6 header of the SRv6 tunnel is set to 255 for all transit IPv4, IPv6, and Ethernet packets encapsulated into SRv6. The hop-limit for OAM packets originated by the CPM on the router is set according to the specific OAM probe.

  The service ingress datapath forwards the packet to one of the SRv6 tunnel candidate egress network IP interfaces based on a hash of the inner packet headers. See Using flow label in load-balancing of IPv6 and SRv6 encapsulated packets for more information about the spraying of service packets over SRv6 tunnel candidate next hops.

14.17.1.2 At the egress PE

1. The procedure in this step is common to the transit router role and the service termination router role.

   On the ingress IP network interface, the SRv6 feature concurrently performs two IPv6 address lookups on a received IPv6 packet:

   - A first (longest prefix match) lookup checks if the address in the outer header DA field matches either an SRv6 local locator subnet, a local service SID, a local End function or a local End.X function. This first lookup is for the current SID.
   - A second lookup is performed on the next SID in the SRH (when the IPv6 packet has an SRH). The SRv6 feature reads next SID using the index value after decrementing the Segments-Left field.

   The subsequent processing depends on the outcome of the first lookup:

   a. If the match is on a local service SID:

- If the payload type is IPv4, IPv6, or Ethernet, the packet is forwarded to the service function processing; see step 2 for more details.

  The payload type refers to the value of the last next-header field in the processing chain of the packet. This could be the next-header field of the outer IPv6 packet, if there is no SRH. This could also be the next-header field of an expired SRH (**Segments-Left** = 0) for which the last SID matches the service SID.

  Note: SR Linux advertises Maximum Segments Left (MSL) `MSD` value in the IS-IS `Maximum Segments Left MSD Type sub-TLV`.

- If the payload type indicates any other protocol, including ICMPv6 (ICMP ping packet) and UDP (potential traceroute message when hop-limit field has a value of 1), the packet is redirected to the CPM for more processing as described in the SR Linux OAM and Diagnostics Guide. Other protocol packets are dropped.

  The CPM generates a specific ICMPv6 message to the address in the SA field of the processed or dropped packet depending on the protocol type and the result of the match of the address in the DA field of the packet. These ICMPv6 reply messages are summarized in Table 45: ICMPv6 reply messages to extracted SRv6 packets.

b. If the match is on a local locator only:

- If the payload type is IPv4, IPv6, or Ethernet, the packet is forwarded to the SRv6 FPE for potential service function processing; see step 2 for more details.
- If the payload type indicates any other protocol, including ICMPv6 (ICMP ping packet) and UDP (potential traceroute message when hop-limit field has a value of 1), the packet is redirected to the CPM for more processing. Protocol matching ICMPv6 ping and UDP traceroute have their packets processed as described in the SR Linux OAM and Diagnostics Guide. Other protocol packets are dropped.

  The CPM generates a specific ICMPv6 message to the address in the SA field of the processed or dropped packet depending on the protocol type and the result of the match of the address in the DA field of the packet. These ICMPv6 reply messages are summarized in Table 45: ICMPv6 reply messages to extracted SRv6 packets.

c. If the match is on a specific local End function and the next SID lookup is not a local locator, the packet is processed as per the transit router role for these functions as detailed in Transit router role with or without segment termination .

d. If the match is on a specific local End function and the next SID resulted in a match on a local locator, the packet is processed as described in step 1.b with the next-header field used in the processing is that of the SRH.

e. If the match is on a specific local End.X function, regardless of the next SID match outcome the packet is processed in accordance with the transit router role for these functions; see Transit router role with or without segment termination for more information.

f. If the match is on a regular IPv6 route or there is no match, the packet is forwarded or dropped. For forwarded packets, the destination address could match the locator prefix or a regular IPv6 prefix of a remote node.

2. The procedure in this step is specific to the service termination router role.

a. When the first lookup match is on a local service SID (service SID encoded in outer DA field), service packet processing is performed inline in the network interface ingress datapath.

- The datapath performs the detailed processing of the specific SID function as per RFC 8986. It then removes the SRv6 encapsulation headers, including SRH if any.
- It decrements and propagates into the IPv4 TTL field or IPv6 hop-limit field of the forwarded inner packet, the minimum of the incoming outer header hop-limit and inner header hop-limit (or TTL) values.
- It performs a lookup of the service SID and forwards the packet to the service context for further processing.

14.17.2 Transit router role in micro-segment SRv6

The datapath behavior in transit SRv6-enabled routers distinguishes the following use cases:

- The DA field of the IPv6 header contains a single SID (shortest path case).
- Multiple SIDs are used in the DA field and eventually in the SRH (LFA repair tunnel or SRv6 policy).

The DA field of the IPv6 header contains a single SID is of the form `2001:db8:00d1:d547::`

where

- `<2001:0db8>` is the SID block
- `<00d1>` is the identifier associated with a specific node
- `<d547>` is an identifier for a specific local service assigned by the node

`<2001:0db8><00d1>` acts both as a locator for the specific node and as a uN function. A packet with this address in the DA field of the IPv6 header is routed toward the specific node with each node along the path matching on `2001:0db8:00d1::/48` and forwarding to the predetermined next hop.

Multiple SIDs in the DA field and eventually in the SRH illustrate the following micro-segment SRv6 behavior.

Suppose that a source node wants to send traffic for a specific service to the destination node D via the nodes B and F. For that example, consider the following:

- The micro-segment ID block is `2001:0db8::/32`.
- The node identifiers (uN) are `0x00b1` for node B, `0x00f1` for node F, and `0x00d1` for node D.
- Node D has advertised `2001:0db8:00d1:d457::` for the specific service.

The source node does the following:

- It constructs the following container: `2001:0db8:00b1:00f1:00d1:d457::`
- It places this container in the DA field of the IPv6 header.
- It forwards the packet to the next-hop determined by an LPM on that address.

On receiving the packet, nodes B and F perform a shift operation (bound to the uN).

Figure 22: Transit routers in micro-segment SRv6 illustrates the preceding example.

Node B matches on `2001:0db8:00b1::/48` (as it has that entry in its FIB) and performs the operation associated with that micro-segment ID:

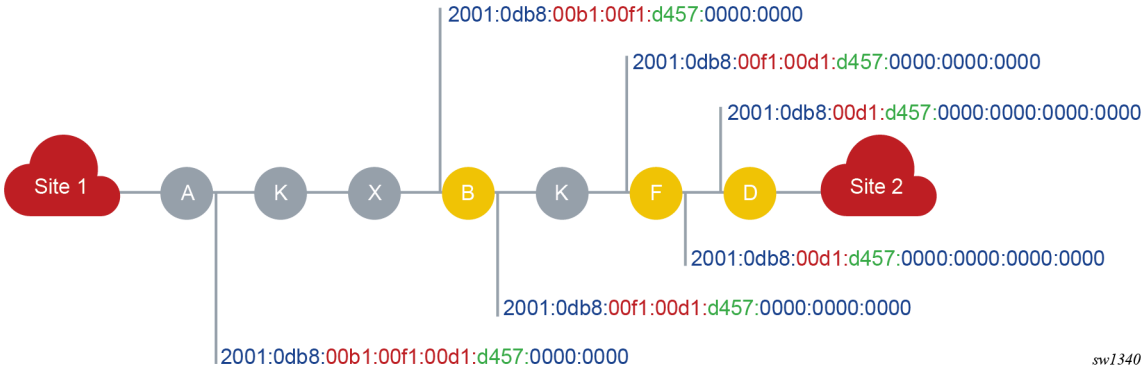
- It removes the 16-bit from the address.
- It shifts the right part toward the MSB.
- It adds a 16-bit block of zeros, the so called End of Container (EoC), as the LSB.

Node B forwards the packet to the next-hop based on a lookup of the resulting DA field.

Node F does the same (based on matching on its own identifier).

Node D does the same but processes the packet in the service corresponding to the identifier.

Figure 22: Transit routers in micro-segment SRv6



It can be that the path is longer than the number of micro-SIDs that can be compressed in 128 bits. In that case, an SRH is used to convey the rest of the path. Each container in the SRH must be of the same form (a block part followed by a sequence of micro-SIDs). At some node along the path, because of the shift operations, the container in the IPv6 DA expires. At that node, micro-segment SRv6 implements a regular SRv6 END function (or END.X function if the match is on a uA).

For example, if node B receives a packet with `2001:0db8:00b1::` in the DA field, it detects that its own identifier is followed by an EoC. Node B copies the next segment or container from the SRH in the DA field.

14.17.3 Transit router role with or without segment termination

The following steps summarize the packet processing for the transit router role. For more information about the specific processing of the SID function see RFC 8986.

1. The procedure in this step is common to the transit router role and the service termination router role.

   On the ingress IP network interface, the SRv6 feature concurrently performs two IPv6 address lookups on a received IPv6 packet:

   - A first (longest prefix match) lookup checks if the address in the outer header DA field matches either an SRv6 local locator subnet, a local service SID, a local End function, or a local End.X function. This first lookup is for the current SID.
   - A second lookup is performed on the next SID in the SRH (when the IPv6 packet has an SRH). The SRv6 feature reads next SID using the index value after decrementing the `Segments-Left` field.

     Note: SR Linux advertises Maximum Segments Left (MSL) `MSD` value in the IS-IS `Maximum Segments Left MSD Type sub-TLV`.

   The subsequent processing depends on the outcome of the first lookup:

   a. If the match is on a local service SID:

      - If the payload type is IPv4, IPv6, or Ethernet, the packet is forwarded to the service function processing; see step 2 in section At the egress PE for more details.

The payload type refers to the value of the last next-header field in the processing chain of the packet. This could be the next-header field of the outer IPv6 packet, if there is no SRH. This could also be the next-header field of an expired SRH (`Segments-Left` = 0) for which the last SID matches the service SID.

- If the payload type indicates any other protocol, including ICMPv6 (ICMP ping packet) and UDP (potential traceroute message when hop-limit field has a value of 1), the packet is redirected to the CPM for more processing. Protocol matching ICMPv6 ping and UDP traceroute have their packets processed as described in the SR Linux OAM and Diagnostics Guide. Other protocol packets are dropped.

  The CPM generates a specific ICMPv6 message to the address in the SA field of the processed or dropped packet depending on the protocol type and the result of the match of the address in the DA field of the packet. These ICMPv6 reply messages are summarized in the table that follows.

Table 45: ICMPv6 reply messages to extracted SRv6 packets

Protocol	Destination IP address match result	ICMPv6 reply(Type/code)
ICMP echo request/reply (See SR Linux OAM and Diagnostics Guide for ICMPv6 Ping support in SRv6)	locator prefix [function | any arg]	echo reply/ping successful
UDP / TCP (See SR Linux OAM and Diagnostics Guide for UDP Traceroute support in SRv6)	locator prefix [function | any arg]	dest unreachable, port unreachable
Any other protocol	locator prefix [function | any arg]	ICMP Parameter Problem/SR Upper-layer Header Error
All protocols including above	locator prefix | unsupported function	dest unreachable, communication prohibited

b. If the match is on a local locator only:

- If the payload type is IPv4, IPv6, or Ethernet, the packet is forwarded to the SRv6 FPE for potential service function processing; see step 2 in section At the egress PE for more information.
- If the payload type indicates any other protocol, including ICMPv6 (ICMP ping packet) and UDP (potential traceroute message when hop-limit field has a value of 1), the packet is redirected to the CPM for more processing. Protocol matching ICMPv6 ping and UDP traceroute have their packets processed as described in the SR Linux OAM and Diagnostics Guide. Other protocol packets are dropped.

  The CPM generates a specific ICMPv6 message to the address in the SA field of the processed or dropped packet depending on the protocol type and the result of the match of the address in the DA field of the packet. These ICMPv6 reply messages are summarized in Table 45: ICMPv6 reply messages to extracted SRv6 packets.

c. If the match is on a specific local End function and the next SID lookup is not a local locator, the packet is processed as per the transit router role for these functions as detailed in step 2 that follows.

d. If the match is on a specific local End function and the next SID resulted in a match on a local locator, the packet is processed as described in the preceding step 1.b with the next-header field used in the processing is that of the SRH.

e. If the match is on a specific local End.X function, regardless of the next SID match outcome, the packet is processed in accordance with the transit router role for these functions; see step 2 that follows.

f. If the match is on a regular IPv6 route or there is no match, the packet is forwarded or dropped. For forwarded packets, the destination address could match the locator prefix or a regular IPv6 prefix of a remote node.

2. The procedure in this step is specific to the transit router role.

If the match is on a local End or End.X SID, the SID termination processing is performed on the packet.

a. If the End or End.X SID is the last SID in the packet encapsulation, meaning there is no SRH or there are only expired SRHs (`Segments-Left` = 0), the packet is sent to the CPM for further processing.

Note: The CPM processes ICMPv6 ping packets and UDP traceroute packets but drops any other protocol type. See the SR Linux OAM and Diagnostics Guide for more information about the processing of ICPMv6 echo request and reply packets and UDP traceroute packets.

b. If the next-header in the IPv6 header is an SRH, the `Segments-Left` field is zero, and the next-header in the SRH is another SRH, the current SRH is removed and the remaining steps are applied on the next SRH.

c. If the `Segments-Left` field is 1 and the SRH mode of the terminated SID is PSP, the SRH is removed. Otherwise, the `Segments-Left` field is decremented and used to read and copy the next SID into the DA field of the outer IPv6 header.

d. Decrement the incoming outer IPv6 header hop-limit and write it into the outer IPv6 header hop-limit field of the outgoing packet.

e. If the first SID lookup of the current SID in the FIB matched an End function, use the outcome of the second SID lookup of the next SID to forward the packet to the next hop of the next SID (in the DA field of the outer IPv6 header).

f. If the first SID lookup of the current SID in the FIB matched an End.X function, override the outcome of the second SID lookup of the next SID with the set of next hops of the adjacency and forward the packet.

g. If both the current and the next SIDs match a local End or End.X SID, the packet is forwarded as indicated in Table 46: Forwarding behavior for back-to-back local SIDs .

Table 46: Forwarding behavior for back-to-back local SIDs

Current SID match	Next SID match	Forwarding action
End	End	**i.** Process Current SID. Terminate Current SID. **ii.** Copy Next SID as IPv6.DA and perform FIB lookup.

Current SID match	Next SID match	Forwarding action
		iii. Process Next SID. Terminate Next SID. **iv.** If Next SID is the last SID, process the inner payload Header. **v.** If the inner payload header is IPv4 or IPv6, perform a FIB lookup to determine the forwarding decision. **vi.** If the inner payload header is TCP/UDP/ICMPv6, extract the packet to CPM.
End	End.X	**i.** Process Current SID. Terminate Current SID. **ii.** Copy Next SID as IPv6.DA and perform FIB lookup. **iii.** Process Next SID. Terminate Next SID. **iv.** If Next SID is the last SID, process the inner payload Header. Otherwise, forward the packet to the L3 adjacency based on Next SID. **v.** If the inner payload header is IPv4 or IPv6, perform a FIB lookup to determine the forwarding decision. **vi.** If the inner payload header is TCP/UDP/ICMPv6, extract the packet to CPM.
End.X	End	**i.** Process Current SID. Terminate Current SID. **ii.** Copy Next SID as IPv6.DA and forward the packet to the L3 adjacency based on the current SID. **iii.** The peer node would send this packet back. **iv.** Process Next SID (this has become the Current SID).

Current SID match	Next SID match	Forwarding action
		v. If Next SID is the last SID, process the inner payload Header. **vi.** If the inner payload header is IPv4 or IPv6, perform a FIB lookup to determine the forwarding decision. **vii.** If the inner payload header is TCP/UDP/ICMPv6, extract the packet to CPM.
End.X	End.X	**i.** Process Current SID. Terminate Current SID. **ii.** Copy Next SID as IPv6.DA and forward the packet to the L3 adjacency based on the current SID. **iii.** The peer node would send this packet back. **iv.** Process Next SID (this has become the Current SID). **v.** If Next SID is the last SID, process the inner payload Header. Otherwise, forward the packet to the L3 adjacency based on Next SID. **vi.** If the inner payload header is IPv4 or IPv6, perform a FIB lookup to determine the forwarding decision. **vii.** If the inner payload header is TCP/UDP/ICMPv6, extract the packet to CPM.

14.17.4 Using flow label in load-balancing of IPv6 and SRv6 encapsulated packets

When traffic is switched via SRv6 shortest path tunnel, flows are distributed across SRv6 tunnels and LAG links of the associated `default` network instance in an ECMP manner. The default hash calculation on the ingress non-default network instance is based on the existing hash procedures of an IPv4, IPv6, or an Ethernet packet. For IPv6 service packets, an option is provided to include the packet's Flow Label field, when not zero, and to hash on the triplet {SA, DA, Flow Label}. The **system load-balancing hash-options ipv6-flow-label** command is used to enable this behavior on an access or network interface.

The ingress non-default network instance copies the output of the hash on the inner packet headers into the flow label field of the outer IPv6 header that it pushes on the SRv6 encapsulated packet. This is regardless of whether the flow label is used or not in the computation of the hash on the service packet.

On a transit router, the hashing of SRv6 encapsulated packets can also use the Flow Label field in the outer IPv6 header to provide more entropy to the load-balancing process of SRv6 packets. The **system load-balancing hash-options ipv6-flow-label** command can be configured on a network facing subinterface to hash on the triplet {SA, DA, Flow Label}. By default, a transit router only hashes on the tuple {SA, DA} in the header of a received IPv6 packet with a non-zero flow label field, including when the packet is SRv6.

14.17.5 Interaction with other datapath features

The following describes the interaction of the SRv6 feature with other datapath features:

- When SRv6 is enabled on default network instance, datapath enables forwarding and receiving SRv6 encapsulated packets on all network facing sub interfaces. The datapath, however, drops an SRv6 encapsulated packet if received from or needs forwarding to a(non default network instance). A service packet received on either a compatible network instance such as `default` network instance or a `ip-vrf` network instance an be encapsulated into an SRv6 packet and forwarded via default network-instance.
- The SRv6 feature performs concurrently a couple of IPv6 address lookups on a packet received with an SRH on a default sub interface. The first lookup is for the current SID in the DA field in the header of the received SRv6 packet and the second is for the next SID in the SRH.

14.18 SRv6 tunnel metric and MTU settings

IGP sets the metric of the SRv6 remote locator prefix route and tunnel or that of a local adjacency SID route to the metric of the computed path of the corresponding route.

The metric of a local End SID route is set to 0, similar to any local route.

The metric of a BGP IPv4/IPv6 or VPN-IPv4/VPN-IPv6 route resolved to a SRv6 tunnel inherits the value of the locator prefix route metric.

You must configure the network interfaces at the ingress PE and at transit P routers with an MTU value that accounts for the fixed IPv6 header (40 bytes) and the additional LFA SRHs (24 bytes each).

In addition, there are network deployments where it is not possible to modify the network interface MTU or to set all network interfaces to the same value. In that case, you must configure the outgoing network interface to reflect this worst case MTU in the network, accounting fixed and variable LFA overhead. The following is the CLI to use for this purpose.

14.19 BGP service control plane extensions

The BGP service control plane required extensions are specified in RFC 9252, *BGP Overlay Services Based on Segment Routing over IPv6 (SRv6)*. BGP requires some changes in the IPv6, VPN-IPv4, VPN-IPv6, and EVPN family routes so that the egress PE can signal the following End programming behaviors to the ingress PE:

- Layer 2 SRv6 service SIDs

  End.DX2 Layer 2 decapsulation and cross-connect to an Epipe egress SAP, signaled by AD per-EVI routes

- Layer 3 SRv6 service SIDs

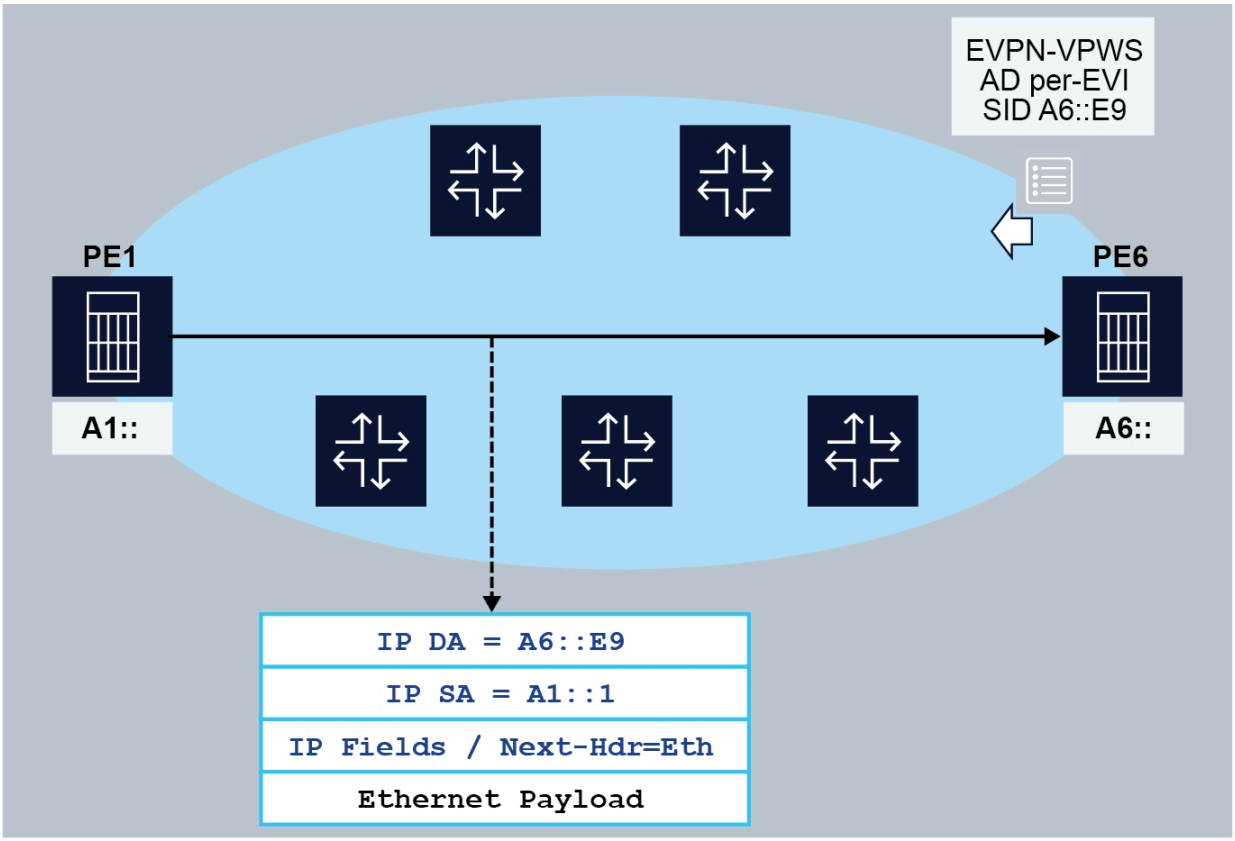
  End.DT4 a IP-VRF (or `default` network instance) route-table lookup, signaled by VPN-IPv4 or EVPN in Interface-less (EVPN-IFL) IPv4 prefix routes (also by IPv4)

  End.DT6 a IP-VRF (or GRT) route-table IPv6 lookup, signaled by VPN-IPv6 or EVPN-IFL IPv6 prefix routes (also by IPv6)

  End.DT46 a IP-VRF route-table lookup for IPv4 or IPv6 prefixes, signaled by VPN-IPv4 or VPN-IPv6 or EVPN-IFL IPv4 or IPv6 prefix routes

The following figure shows an example for the End.DX2 behavior for EVPN-VPWS services.

Figure 23: End.DX2 behavior for EVPN-VPWS



sw4084

The ingress and egress PEs behave as follows:

- The egress PE (PE6) advertises an A-D per EVI route with the SRv6 Service SID that identifies the End.DX2 behavior. The service SID includes the configured locator in the Epipe (A6::), as well as the allocated function (E9), which identifies the Epipe at the egress PE.

- The ingress PE (PE1) imports the A-D per EVI route and creates an EVPN destination in the corresponding Epipe to A6::E9.
- When PE1 receives frames at the non-default network.instance, it encapsulates the frames into an SRv6 packet using the configured IP SA. The IP DA is the EVPN destination SID.
- Shortest path forwarding is considered in the example shown in Figure 23: End.DX2 behavior for EVPN-VPWS, and therefore the EVPN destination SID is encoded in the IP DA. If TI-LFA is required, PE1 modifies the encapsulation to include an SRH and additional SIDs.
- When the SRv6 packet arrives at PE6, the SID encoded in the IP DA identifies the packet for termination on PE6 and the Epipe for decapsulation and forwarding.

Similar procedures are followed for the other required services.

The following table lists the functions that micro-segment SRv6 implements to support the requirements.

Table 47: Micro-segment SRv6 functions

SID function endpoint behavior	Codepoint	SID type: End.SID	SID type: End.X SID	SID type: LAN End.X SID	Advertising protocol	Supported
uDT6	62	Yes	No	No	BGP or static	Yes[5]
uDT4	63	Yes	No	No	BGP or static	Yes[5]
uDT46	64	Yes	No	No	BGP or static	Yes[5]
uDX2	65	Yes	No	No	BGP or static	Yes[5]

14.19.1 BGP extensions

The following BGP extensions are supported, as defined in RFC 9252:

- The following SRv6 Service TLVs:
  - SRv6 Service TLV encoded in the BGP Prefix-SID attribute
  - SRv6 SID Information Sub-TLV (SRv6 Service Sub-TLV type 1) encoded in the SRv6 Service TLV
  - SRv6 SID Structure Sub-Sub-TLV (SRv6 Service Data Sub-Sub-TLV type 1)
- Transposition of 16 or 20 bits of the FUNCTION to the Label field of the NLRI
- Arg.FE2 arguments used for split-horizon filtering in EVPN multihoming, advertised in the EVPN AD per-ES routes with 16 bits of the argument transposed to the label field of the ESI Label extended community.

The BGP extensions are applied to the following routes by setting the behavior field in the SRv6 Services TLV, as defined in RFC 8986:

- VPN-IPv4
- VPN-IPv6

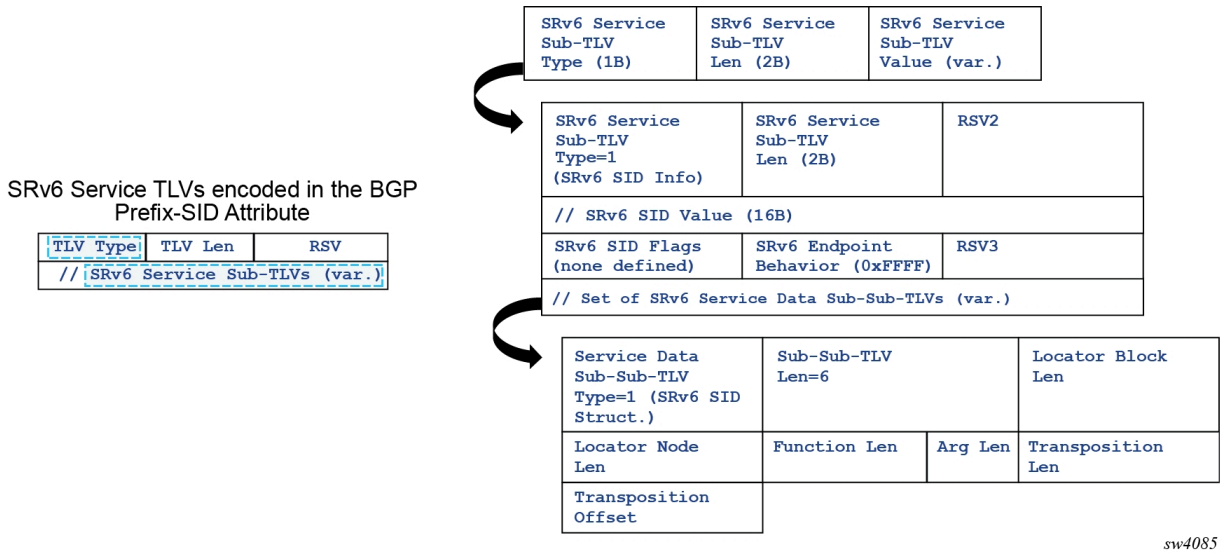
[5] BGP advertises the supported endpoint behaviors and accepts any behavior codepoint with a supported NLRI type.

- IPv4
- IPv6
- EVPN AD per-EVI
- EVPN AD per-ES

14.19.2 Advertising SRv6 service TLVs

The EVPN, VPN-IPv4, VPN-IPv6, IPv4, and IPv6 routes for the SRv6-enabled services are advertised along with the SRv6 Service TLV. The TLV format is described in RFC 9252 and shown in the following figure.

Figure 24: SRv6 service TLV format



The SRv6 Service TLV encoded in the BGP Prefix-SID attribute can have two different types:

- Type 5 is used for Layer 3 service SIDs or the SIDs signaled for IP-VRF services with VPN-IP or EVPN-IFL routes. Layer 3 service SIDs are also supported for the base router along with IPv4 or IPv6 routes.
- Type 6 is used for Layer 2 service SIDs or the SIDs signaled for VPWS (or MAC-VRF) services. Type 6 is supported along with AD per-EVI and per-ES routes for Epipes.

Note: MAC-VRF is not supported in SR Linux.

The SRv6 Service TLV may contain an unordered list of sub-TLVs, but currently the SRv6 Service TLV is advertised with only one sub-TLV: the SRv6 SID Info sub-TLV (type 1). This sub-TLV encodes the following information:

- For the SID value, the entire 128-bit SID is allocated to the service, including the locator configured for the service and the allocated FUNCTION (which can be dynamically allocated or statically configured on the service). The ARGUMENT is always 0.
- SID flags are all zero.

- Endpoint behavior encodes the behavior as in RFC 8986, in decimal values. The following values are relevant for SR Linux:
  - 18 – End.DT6
  - 19 – End.DT4
  - 20 – End.DT46
  - 21 – End.DX2
  - 24 – End.DT2m (only for AD per-ES routes)
- One SID Structure Sub-Sub-TLV (Service Data Sub-Sub-TLV type 1)

The SID Structure Sub-Sub-TLV is always included in routes with label fields and always uses the following values when advertised:

- Locator Block Length - encodes the length of the block configured in the locator for the service, default value of 32
- Locator Node Length - the length of the node configured in the locator for the service, default value of 16
- Function Length - default value of 16
- Argument Length - 0 (default) or 16 (configured)
- Transposition Length (TL) - 20 for EVPN and VPN-IP routes with full SIDs and 16 for micro segments and 0 for IP routes in the base router
- Transposition Offset (TO)
  - for EVPN and VPN-IP routes:
    - If the Function Length equals 20 or 16, the Transposition Offset (TO) value equals the prefix length configured in the locator
    - If the Function Length is greater than 20, the TO value equals:

      (prefix length configured in the locator) + (FunctLength 20)
  - for IP routes in base router, TO value is always 0
- For IP routes in the base router, the TO value is always 0.

14.19.3 Configuring a BGP global SRv6 source address

Procedure

To encapsulate SRv6 within a BGP IPv6 unicast session, you must first configure the address families and establish the BGP IPv6 unicast session, and then use the **network-instance.protocols.bgp.srv6.source-address** command to specify the SRv6 source address for encapsulation.

Example

In the following configuration example, the BGP instance includes an SRv6 block that sets the source address 2001:db8:100::1. The router (10.0.0.1) uses the IPv6 address 2001:db8:100::1 as the source address for all BGP IPv6-unicast sessions, enabling the BGP session to be carried over the SRv6 infrastructure.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp
```

```
   network-instance default {
        protocols {
            bgp {
                admin-state enable
                autonomous-system 65001
                router-id 10.0.0.1
                afi-safi ipv6-unicast {
                    admin-state enable
                }
                srv6 {
                    source-address 2001:db8:100::1
                }
            }
        }
    }
```

14.19.4 Enabling SRv6 TLVs for `IPv4-unicast` BGP routes

Procedure

You can enable SRv6 TLVs for IPv4-unicast BGP routes by using the command **network-instance protocols bgp afi-safi ipv4-unicast ipv4-unicast srv6 add-srv6-tlvs**.

Example

```
--{ + running }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast ipv4-
unicast srv6
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    ipv4-unicast {
                        srv6 {
                            add-srv6-tlvs {
                                locator-name srl_loc1
                            }
                        }
                    }
                }
            }
        }
    }
```

14.19.5 Enabling SRv6 TLVs for `IPv6-unicast` BGP routes

Procedure

You can enable SRv6 TLVs for IPv6-unicast BGP routes by using the command **network-instance protocols bgp afi-safi ipv6-unicast ipv6-unicast srv6 add-srv6-tlvs**.

Example

```
--{ + running }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv6-unicast ipv6-
unicast srv6
    network-instance default {
```

```
        protocols {
            bgp {
                afi-safi ipv6-unicast {
                    ipv6-unicast {
                        srv6 {
                            add-srv6-tlvs {
                                locator-name srl_loc1
                            }
                        }
                    }
                }
            }
        }
    }
```

14.19.6 Processing SRv6 TLVs for `IPv4-unicast` BGP routes

Procedure

To process received SRv6 TLVs for IPv4-unicast BGP routes, set the parameter **process-received-srv6-tlvs** within the command **network-instance protocols bgp afi-safi ipv4-unicast ipv4-unicast srv6** context to `true`.

Example

```
--{ +* candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp afi-safi ipv4-unicast ipv4-
unicast srv6
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv4-unicast {
                    ipv4-unicast {
                        srv6 {
                            process-received-srv6-tlvs true
                            add-srv6-tlvs {
                                locator-name srl_loc1
                            }
                        }
                    }
                }
            }
        }
    }
```

14.19.7 Processing SRv6 TLVs for `IPv6-unicast` BGP routes

Procedure

To process received SRv6 TLVs for IPv6-unicast BGP routes, set the parameter **process-received-srv6-tlvs** within the command **network-instance protocols bgp afi-safi ipv6-unicast ipv6-unicast srv6** context to `true`.

Example

```
--{ + running }--[  ]--
```

```
# info with-context network-instance default protocols bgp afi-safi ipv6-unicast ipv6-
unicast srv6
    network-instance default {
        protocols {
            bgp {
                afi-safi ipv6-unicast {
                    ipv6-unicast {
                        srv6 {
                            process-received-srv6-tlvs true
                            add-srv6-tlvs {
                                locator-name srl_loc1
                            }
                        }
                    }
                }
            }
        }
    }
```

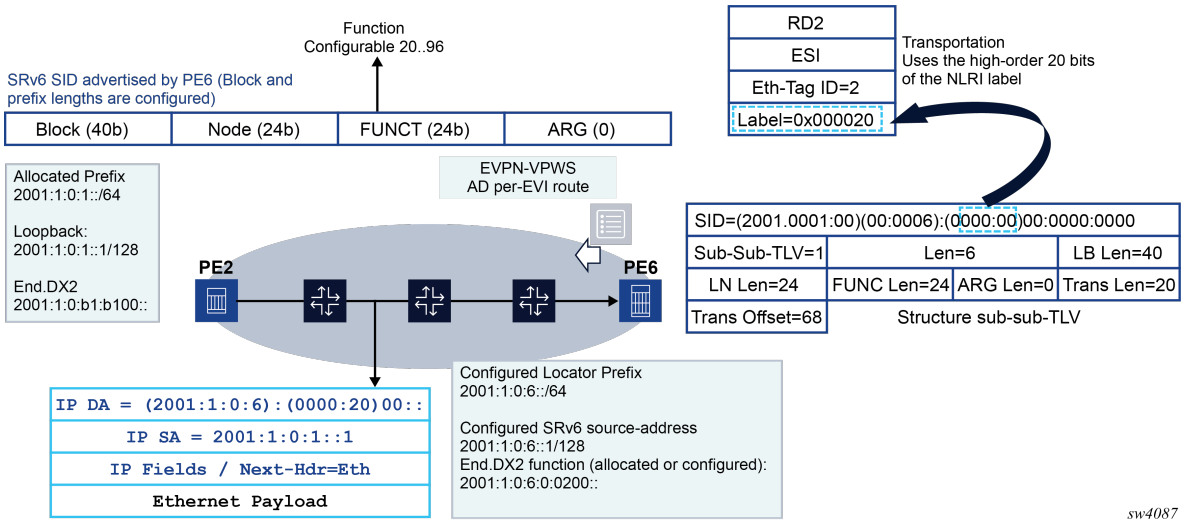
14.19.8 Transposition procedures when advertising service routes

The purpose of the SID Structure Sub-Sub-TLV is twofold:

- Advertise the structure of the SRv6 SID used in the service, including the length of the Locator block, node, function and argument.
- Support transposition procedures for efficient service route packing. The FUNCTION is transposed into the label field in the route's NLRI. Because the rest of the SID is common for routes of the same type in the service, this transposition operation supports efficient packing of routes into the same BGP update.

The following figure shows how the FUNCTION part of the SID is transposed. This example illustrates how transposition works for EVPN-VPWS, and it would be similar for VPN-IP routes.

Figure 25: Transposition of the FUNCTION into the NLRI



In the preceding figure, PE6 is configured with an Epipe that uses a configured locator with LB length = 40 bits and LN length = 24 bits. The Function length is set at 24, and 20 bits are always transposed into the NLRI (non-configurable). In the preceding figure, the following rules apply:

- On reception, the router can build any SID out of the received route, irrespective of transposition, as long as the lengths are correctly encoded.
- On transmission, the system performs a transposition for VPN-IP and EVPN service routes as follows:
- The following rules apply for Function Length:
  - If Function Length is greater than 20 in the Locator configuration, the function bits are put at the right-most bits of the L bits. For example, if LB LEN is 40 bits and the LN Len is 24 bits:
  - If Function Length = 20, the entire function is transposed into the label field, and the following is signaled in the route:

    Length [LBL, LNL, FL, AL] : [40, 24, 20, 0]

    TL:20, TO:64

14.19.9 Supported service routes for SRv6

The supported service routes for SRv6 are:

- IPVRF services, configured for SRv6:
  - VRF -IPv4 routes
  - VRF -IPv6 routes
- Base router
  - IPv6 routes, if configured for SRv6 in IPv6 family
  - IPv4 routes, if configured for SRv6 in IPv4 family
- Epipe services, configured for SRv6:
  - EVPN AD per-EVI routes
  - EVPN AD per-ES routes, for VPWS that make use of Ethernet Segments

14.19.10 Removing SRv6 TLVs from `IPv4-unicast` routes for a BGP `peer-group`

Procedure

You can configure a BGP group to remove any SRv6 TLVs from received IPv4-unicast updates by setting the parameter **strip-srv6-tlvs** within the command **network-instance protocols bgp group afi-safi ipv4-unicast ipv4-unicast srv6** context to `true`.

Example

```
--{ + running }--[  ]--
# info with-context network-instance default protocols bgp group group_1 afi-safi ipv4-
unicast ipv4-unicast srv6
    network-instance default {
        protocols {
            bgp {
                group group_1 {
                    afi-safi ipv4-unicast {
                        ipv4-unicast {
                            srv6 {
                                strip-srv6-tlvs true
                            }
```

```
                        }
                    }
                }
            }
        }
    }
```

14.19.11 Removing SRv6 TLVs from `IPv6-unicast` routes for a BGP `peer-group`

Procedure

You can configure a BGP group to remove any SRv6 TLVs from received IPv6-unicast updates by setting the parameter **strip-srv6-tlvs** within the command **network-instance protocols bgp group afi-safi ipv6-unicast ipv6-unicast srv6** context to `true`.

Example

```
--{ + running }--[  ]--
# info with-context network-instance default protocols bgp group group_1 afi-safi ipv6-
unicast ipv6-unicast srv6
    network-instance default {
        protocols {
            bgp {
                group group_1 {
                    afi-safi ipv6-unicast {
                        ipv6-unicast {
                            srv6 {
                                strip-srv6-tlvs true
                            }
                        }
                    }
                }
            }
        }
    }
```

14.19.12 Discarding BGP learned SRv6 routes

Procedure

You can discard any BGP learned routes that carry SRv6 TLVs using the command **network-instance protocols bgp neighbor srv6 drop-routes-with-srv6-tlvs**.

Example: Discarding the BGP learned SRv6 routes

In the following configuration example, the router is set up to drop any BGP-learned routes that carry SRv6 TLVs for a specific neighbor (10.1.1.2) that belongs to the EBGP-UNDERLAY peer-group.

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance default protocols bgp neighbor 10.1.1.2
    network-instance default {
        protocols {
            bgp {
                neighbor 10.1.1.2 {
                    peer-group EBGP-UNDERLAY
                    srv6 {
                        drop-routes-with-srv6-tlvs true
```

```
                    }
                }
            }
        }
    }
```

14.19.13 BGP next hop for SRv6 service routes

As specified in RFC 9252, *BGP Overlay Services Based on Segment Routing over IPv6 (SRv6)*, the egress PE may set the next hop to any of its IPv6 addresses. When the IPv6 address value is not covered by the SRv6 locator from which the SRv6 Service SID is allocated, the ingress PE performs reachability checks for the SRv6 Service SID in addition to the BGP next-hop reachability procedures.

Next hop and locator resolution considerations include the following:

- On reception of a BGP SRv6 service route, both the locator and the next hop are resolved independently in the route table.
- For base instance routes (not service routes), the **network-instance protocols bgp afi-safi ipv6-unicast ipv6-unicast srv6 process-received-srv6-tlvs true** command triggers the independent resolution of the next hop and the locator reachability (and collates their states that drive route programming). The locator state is considered only if this command is configured.
- Whether the router performs **network-instance protocols bgp group next-hop-self**, there is no effect on the RIB-IN processing and reachability because the next-hop behavior is a RIB-OUT parameter.
- If a received route has a resolved next hop but unresolved locator, the **show network-instance default protocols bgp route** command shows "valid/best" but not "used" in the route flags.

14.19.14 Route policy support for matching and modifying BGP SRv6 service routes

Route-matching criteria and route-modifying actions that are specific to BGP routes carrying SRv6 TLVs can be configured for route policies.

Note: SRv6 TLV match criteria and modifying actions are supported only for policies configured with IPv4, IPv6, VPN-IPv4, VPN-IPv6, and EVPN route families. If the policy is also configured with other route families, the routes do not match to the SRv6 TLV criteria, regardless of whether the **routing-policy policy statement match bgp srv6 tlv** command is configured to **present** or **not present** .

14.19.14.1 Specifying SRv6 TLV match criteria

Procedure

You can specify SRv6 TLV presence as a match criterion for BGP route using the command **routing-policy policy statement match bgp srv6 tlv**.

If **srv6 tlv** is configured as `present`, the policy entry matches only those BGP routes that carry the SRv6 TLVs. When configured as `not present`, the policy entry matches only those BGP routes that do not include the SRv6 TLVs.

Example

In the following configuration example, the policy entry matches only those BGP routes that carry the SRv6 TLVs.

```
--{ +* candidate shared default }--[  ]--
# info with-context routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    bgp {
                        srv6 {
                            tlv present
                        }
                    }
                }
            }
        }
    }
```

14.19.14.2 Specifying SID prefix match criteria for BGP routes

Procedure

You can specify SRv6 TLV presence as a match criterion for BGP route using the command **routing-policy policy statement match bgp srv6 sid-prefix**.

When a **sid-prefix** is configured, the policy entry matches only those BGP routes whose associated SID prefix equals or falls within the defined prefix range.

Example

In the following configuration example, the policy entry matches only those BGP routes that have an SRv6 SID (or uSID) within the prefix`2001:db8:100::/48`.

```
--{ +* candidate shared default }--[  ]--
# info with-context routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    bgp {
                        srv6 {
                            sid-prefix 2001:db8:100::/48
                        }
                    }
                }
            }
        }
    }
```

14.19.14.3 Specifying SRv6 locator action in a routing policy

Procedure

You can specify the SRv6 locator action to BGP routes that match a policy statement.

Note: The default routing policy action (**routing-policy default-action bgp**) is SRv6 locator (**srv6 locator**).

Example

The following configuration identifies a BGP policy (`policy01`) matching routes with an SRv6 TLV and applies the SRv6 locator `srl_loc1` action to guide SRv6 traffic for those routes.

```
--{ + running }--[  ]--
# info with-context routing-policy policy policy01
    routing-policy {
        policy policy01 {
            statement 100 {
                match {
                    bgp {
                        srv6 {
                            tlv present
                        }
                    }
                }
                action {
                    bgp {
                        srv6 {
                            locator srl_loc1
                        }
                    }
                }
            }
        }
    }
```

14.19.15 BGP Link State (BGP-LS) extensions for SRv6

For information about BGP-LS extensions for SRv6 supported in SR Linux, see the "BGP Link-State" section of the *SR Linux Routing Protocols Guide.*

14.20 Service extensions

The following section describes the service extensions for originating and terminating SRv6 services and applies to both regular and micro-segment SRv6, unless stated otherwise.

14.20.1 SRv6 IP-VRF services

SRv6 for IP-VRF allows the transport of IP-VRF-related IPv4 and IPv6 data across an SRv6-enabled network. To this end, IP-VRF-related data is sent to an ingress SRv6 router, where it is encapsulated and forwarded via an SRv6 tunnel. The SRv6 tunnel transports the encapsulated data across the SRv6-enabled network to an egress SRv6 router, where it is decapsulated and forwarded further as IP-VRF-related data. SRv6-tunneled data is encapsulated using an IPv6 header, where the destination address is a unique SRv6 segment identifier (SID), and is processed and forwarded in the IPv6 data plane.

An SRv6 SID is a preconfigured 128-bit routable IPv6 prefix address that is encoded in three parts: a locator, a function, and an argument. The locator is a summary IPv6 prefix for a set of SRv6 SIDs

instantiated on an SRv6-capable router. It is used to route the data within the IPv6 transport network. Each participating SRv6-capable router needs its unique locator, based on a common block that all participating SRv6-capable routers share in the IPv6 address space. The function is an opaque identifier that indicates the local behavior at the endpoint of an SRv6 segment. The focus in this topic is on the SRv6 End.DT4 and the SRv6 End.DT6 functions for the IP-VRF, performing a prefix lookup in the IP-VRF service IPv4 route table (End.DT4) or in the IP-VRF service IPv6 route table (End.DT6). The argument is not used in SR Linux and is set to all zeros.

The local router installs its locator prefix in its IPv6 route table and forwarding information base (FIB), and advertises its locator prefix in IS-IS with the SRv6 locator sub-TLV. Each remote router populates its route table and FIB with the received locator prefixes, including the tunneled next hop to the originating router. Each remote router also populates its IP-VRF service route table with the received network prefixes, including the tunneled next hop to the IP-VRF of the originating router.

14.20.1.1 Configuring IP-VRF for SRv6

Procedure

IP-VRF services support SRv6 End.DT4, End.DT6, and End.DT46 behaviors. IP-VRFs support IPv4 and IPv6 routes that are advertised in the VPN-IPv4 and VPN-IPv6 families, as well as EVPN IP prefix routes in Interface-less mode.

Example

In the following configuration example, an IP-VRF instance `IPVRF1` is created and the SRv6 segment-routing instance 1 is defined with locator `srl_loc1`. The BGP-IPVPN instance 1 is then configured to use that SRv6 instance (`default-locator srl_loc1 instance 1`) with source address 2001:db8:100::1.

```
--{ + running }--[  ]--
# info with-context network-instance IPVRF1 protocols bgp-ipvpn bgp-instance 1 srv6
    network-instance IPVRF1 {
        protocols {
            bgp-ipvpn {
                bgp-instance 1 {
                    srv6 {
                        default-locator srl_loc1
                        instance 1
                        source-address 2001:db8:100::1
                    }
                }
            }
        }
    }
```

14.20.2 EVPN VPWS services with SRv6

Service providers prefer an optimized, standardized, and unified control plane for VPNs. EVPN-VPWS is supported in SRv6 networks that may also run other EVPN-based services, such as EVPN-based VPLS services or Layer 3 EVPN IFL (interface-less) services. From a control plane perspective, EVPN-VPWS is a simplified point-to-point version of RFC 7432, because there is no need to advertise MAC/IP advertisement routes in VPWS. EVPN-VPWS is described in RFC 8214, and the signaling aspects to support SRv6 are specified in RFC 9252.

EVPN-VPWS supports all-active multihoming (per-flow load-balancing multihoming) as well as single-active multihoming (per-service load-balancing multihoming), using the same Ethernet segments (ESs) used for EVPN-based VPLS services. EVPN-VPWS uses route type 1 and route type 4; it does not use route types 2, 3, or 5, because MAC/IP routes, inclusive multicast routes, or IP-prefix routes are not required.

EVPN-VPWS uses AD per-EVI routes, and optionally, if multihoming is used, AD per-ES and ES routes are required:

- route type 1 - Auto-discovery per EVPN instance (AD per-EVI). This route type is used in all EVPN-VPWS scenarios, with or without multihoming. For EVPN-VPWS, the Ethernet tag field is encoded with the local attachment circuit (AC) of the advertising PE. This value is configured using the **network-instance protocols bgp-evpn bgp-instance vpws-attachment-circuits local local-attachment-circuit ethernet-tag** command. The route distinguisher (RD), label, and the Ethernet segment identifier (ESI) are encoded as for EVPN-based VPLS. The label field is used as service label. In case of multihoming, AD per-EVI routes containing the same ESI are used to provide aliasing and a backup path to the PEs part of the ES. The L2 MTU field is encoded with the service MTU configured in the Epipe. The flags used for EVPN-VPWS are:
  - Flag C: this flag is set if a control word is configured in the service; however, this does not apply if the transport is SRv6.
  - Flag P: this flag is set if the advertising PE is a primary PE.
    - If no multihoming is used, there is no primary PE (P = 0).
    - In all-active multihoming, all PEs in the ES are primary (P = 1).
    - In single-active multihoming, only one PE per-EVI in the ES is a primary (P = 1).
  - Flag B: this flag is set if the advertising PE is a backup PE.
    - Flag B is only set in case of single-active multihoming and only for one PE, even if more than two PEs are present in the same single-active ES. The backup PE is the winner of the second designated forwarder (DF) election (excluding the DF). The remaining non-DF PEs send B = 0.

  If there is no multihoming, the ESI, flag P, and flag B are set to zero.
- route type 1 - Auto-discovery per Ethernet segment (AD per-ES). This route type has the same encoding as for EVPN-based VPLS. The AD per-ES route is only used in multihoming scenarios where it is advertised from the PE for each ES. This route type carries the ESI label (used for split-horizon, but only for VPLS services and not for Epipe services) and can affect procedures such as the DF election, as well as the aliasing on remote PEs.
- route type 4 - ES route. This route type has the same encoding as for EVPN-based VPLS. The ES route is only used in multihoming scenarios. This route type advertises a local configured ES. The exchange of this route type can discover remote PEs that are part of the same ES and the DF election algorithm among them.

14.20.2.1 Configuring EVPN VPWS services with SRv6

Procedure

To configure EVPN-VPWS over SRv6, use the command **network-instance protocols bgp-evpn bgp-instance vpws-attachment-circuits local local-attachment-circuit srv6**.

Example

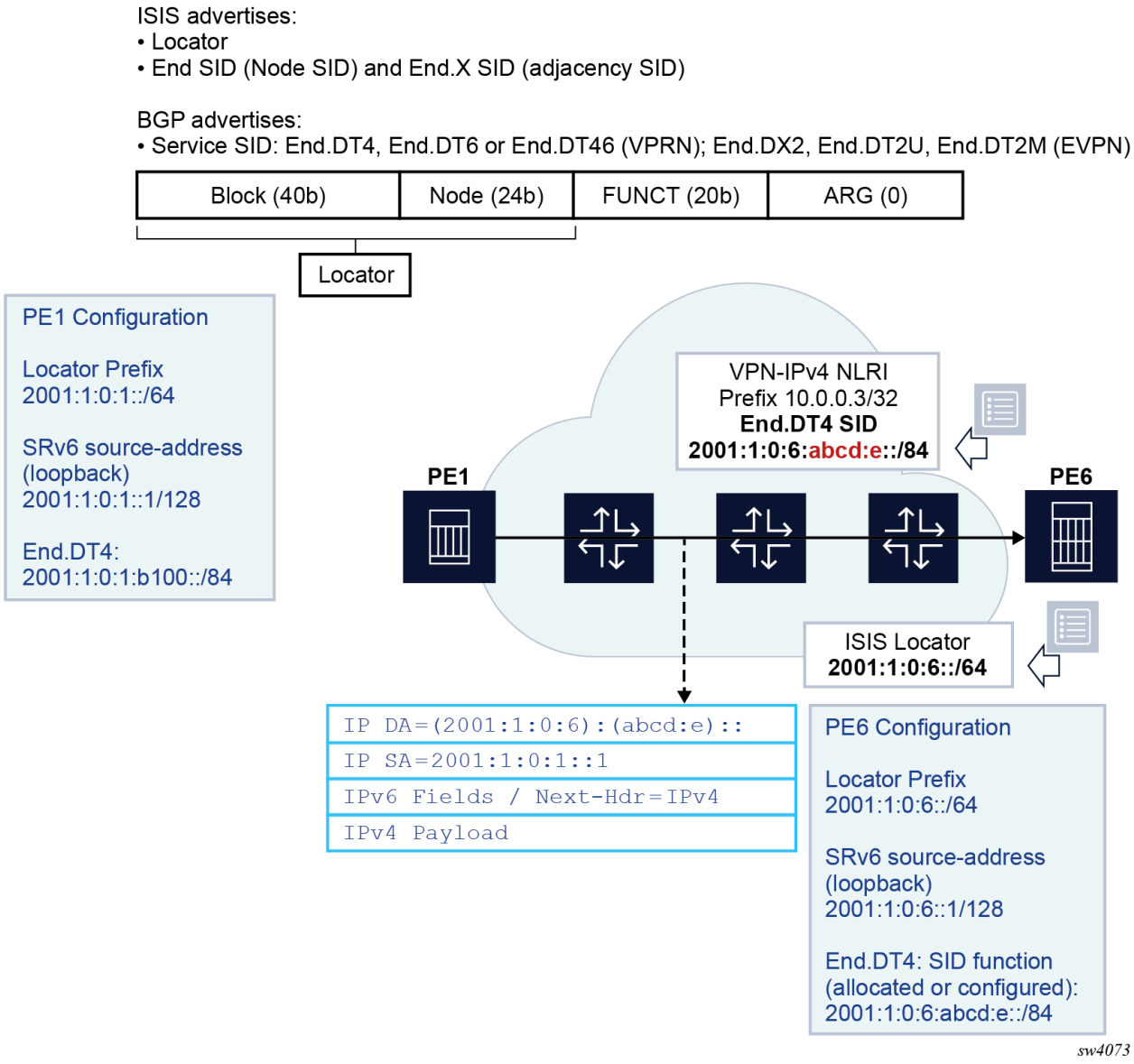
The following example configures a EVPN VPWS service with SRv6

```
--{ + candidate shared default }--[  ]--
# info with-context network-instance vpws_srl
    network-instance vpws_srl {
        type vpws
        admin-state enable
        interface ethernet-1/32.1 {
            connection-point cp1
        }
        protocols {
            bgp-evpn {
                bgp-instance 1 {
                    admin-state enable
                    encapsulation-type srv6
                    evi 1
                    vpws-attachment-circuits {
                        local {
                            local-attachment-circuit ac_1 {
                                ethernet-tag 1
                                connection-point cp2
                                srv6 {
                                    default-locator srl_loc1
                                    instance 1
                                }
                            }
                        }
                        remote {
                            remote-attachment-circuit ac_1 {
                                ethernet-tag 1
                                connection-point cp2
                            }
                        }
                    }
                    srv6 {
                        source-address 2001:db8:100::1
                    }
                }
            }
        }
        connection-point cp1 {
        }
        connection-point cp2 {
        }
        segment-routing {
            srv6 {
                instance 1 {
                    locator srl_loc1 {
                        full-segment {
                            function {
                                end-dx2 {
                                    value 800
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

14.21 SRv6 data and control plane operation

The following figure shows the operation of the data and control planes when an IP-VPN route is resolved to an SRv6 tunnel.

Figure 26: SRv6 data and control plane operation



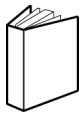
The PE6 egress router advertises the locator route that contains its locator prefix and, optionally, a local node SID (End) in IS-IS. It also advertises the SID of each adjacency (End.X) to its IS-IS neighbors.

The locator prefix provides the route to reach PE6 and is used by other routers to forward an SRv6 packet destined for any SID owned by PE6. Other routers use the End and End.X SIDs to create the repair tunnel for the Remote LFA and TI-LFA backup paths.

In BGP, PE6 advertises a VPN-IPv4 route and includes the End.DT4 service SID, which is equivalent to the SR-MPLS service label in the label per-VRF model. Unlike the SR-MPLS service label, the SRv6 End.DT4 SID contains both the function value that identifies the specific VRF-ID in PE6 and the locator prefix that provides the reachability to router PE6.

The PE1 router resolves the received VPN-IPv4 route by validating the next hop and checking the reachability of the locator prefix of PE6 in the routing table. When PE1 receives an IPv4 packet from a CE node, it pushes an outer IPv6 header that contains the End.DT4 SID in the DA field and looks up the address in the routing table. The packet is then forwarded to one of the next hops of the route of the locator prefix of PE6.

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback