



Nokia Service Router Linux 7250 Interconnect Router 7730 Service Interconnect Router Release 25.3

Segment Routing Guide

3HE 21402 AAAA TQZZA
Edition: 01
March 2025

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.

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia's customers ("You"/"Your") in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia's site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Copyright and trademark: Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

© 2025 Nokia.

Table of contents

1	About this guide.....	6
1.1	Precautionary and information messages.....	6
1.2	Conventions.....	6
2	What's new.....	8
3	About segment routing.....	9
3.1	IS-IS extensions for segment routing.....	10
3.1.1	Router Capability TLV advertisement.....	11
3.1.2	Prefix SID processing.....	11
3.1.3	Adjacency SID processing.....	14
3.1.4	Datapath programming by SID type.....	15
3.2	Supported functionality.....	16
4	SR-MPLS configuration on the default network instance.....	18
4.1	Defining the SRGB and enabling SR-MPLS.....	18
4.2	Defining protocol-independent prefix SIDs.....	19
5	SR-MPLS configuration on the IS-IS instance.....	21
5.1	Configuring an IS-IS node SID.....	21
5.2	Dynamic adjacency SIDs configuration.....	22
5.2.1	Defining the dynamic SRLB.....	22
5.2.2	Enabling dynamic adjacency SID assignment for the IS-IS instance.....	23
5.3	Static adjacency SIDs configuration.....	24
5.3.1	Defining the static SRLB.....	24
5.3.2	Configuring static adjacency SID assignment for the IS-IS instance.....	25
5.3.3	Configuring static adjacency SIDs for an interface.....	26
5.4	Overriding adjacency SID assignment mode on an interface.....	26
5.5	Enabling SR-MPLS on the IS-IS instance.....	28
6	Loop-free alternates.....	29
6.1	Remote LFA with segment routing.....	29
6.1.1	Remote LFA PQ node algorithm.....	29
6.1.2	Label stack encoding.....	31

6.1.3	Remote LFA rules and limitations.....	32
6.1.4	Remote LFA node-protect operation.....	32
6.2	Configuring Remote LFA.....	34
6.2.1	Excluding interfaces from LFA.....	35
6.2.2	Excluding an IS-IS instance from LFA.....	35
6.2.3	Configuring node protection in RLFA.....	36
6.3	Topology-independent LFA.....	36
6.3.1	TI-LFA algorithm.....	37
6.3.2	TI-LFA feature interaction and limitations.....	38
6.3.3	LFA protection option applicability.....	39
6.3.4	TI-LFA node-protect operation.....	39
6.3.4.1	TI-LFA and remote LFA node protection feature interaction and limitations.....	42
6.4	Configuring TI-LFA.....	42
7	BGP shortcuts configuration over segment routing tunnels.....	43
7.1	Configuring BGP shortcuts over segment routing.....	43
8	Segment routing display commands.....	45
8.1	Displaying the SID database.....	45
8.2	Displaying label block information.....	47
8.3	Displaying tunnel table entries.....	47
9	LSP ping and trace for segment routing tunnels.....	51
10	Traffic Engineering.....	52
10.1	Configuring TE identifier.....	52
10.2	Configuring TE interface.....	53
10.3	Configuring TE metric.....	53
10.4	Configuring TE admin-groups.....	54
10.5	Configuring Shared Risk Link Groups and SRLG membership.....	55
10.6	Advertising link delay with IS-IS.....	56
11	TE-Policy.....	59
11.1	Uncolored SR-MPLS TE-Policy.....	59
11.1.1	Basic concepts.....	59
11.1.2	Binding segments.....	60
11.1.3	Configuring SR-MPLS TE policy.....	61

11.1.4	Configuring BSID to a static MPLS label range.....	63
11.1.5	Configuring BSID to a MPLS label value.....	64
11.1.6	SR-MPLS TE policy path computation.....	65
11.1.6.1	Explicit path.....	65
11.1.6.2	Path Computation Element Protocol (PCEP).....	67
11.1.6.3	Local CSPF (Constrained Shortest Path First) based path computation.....	86
11.1.6.4	Path computation fallback.....	89
11.1.7	Uncolored SR-MPLS TE-policy tags.....	91
11.1.7.1	Associating an SR-MPLS TE policy with a tag-set.....	92
11.1.7.2	Associating a tag-set with BGP next-hop resolution.....	92
11.1.7.3	Configuring a tag as mandatory for BGP route resolution.....	93
11.1.8	Uncolored SR-MPLS TE path failure codes.....	93
11.2	Colored TE policy.....	95
11.2.1	Basic concepts.....	96
11.2.2	Best path selection process.....	96
11.2.3	Statically-configured colored TE policies.....	97
11.2.3.1	Configuring a static local colored TE policy.....	97
11.2.3.2	Configuring a static non-local colored TE policy.....	98
11.2.4	BGP signaled colored TE policy.....	99
11.2.4.1	Importing a static non-local colored TE policy.....	100
11.2.4.2	Colored TE policy encoding using TLV.....	101
11.2.4.3	Colored TE policy traffic steering.....	101
11.2.4.4	Colored TE policy data forwarding.....	103
11.2.4.5	Next-hop address encoding for colored TE policy.....	103

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

Nokia SR Linux documentation uses the following command conventions.

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

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

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

2 What's new

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

Table 1: What's new in Release 25.3.1

Topic	Location
PCE Associations	PCEP Associations
Path computation fallback	Path computation fallback
Support for Colored TE policy	Colored TE policy

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 flexibility 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 TLVs 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 2: TLV 242 sub-TLVs

Sub-TLVs	Description
SR Capabilities sub-TLV	Advertises one SRGB block and data plane capability: <ul style="list-style-type: none"> • 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 3: 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 <code>system0.0</code> , 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 a 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, a 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 4: 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 5: Datapath programming by SID type

SID type	Datapath programming
IS-IS node SID	<p>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.</p> <p>The POP operation indicates that the default network instance is the context for the IP header lookup that follows.</p>
Protocol-independent prefix SID	<p>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.</p> <p>The POP operation indicates that the default network instance is the context for the IP header lookup that follows.</p>
IS-IS adjacency SID (P2P and LAN)	<p>When advertised, the adjacency SID:</p> <ul style="list-style-type: none"> 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. <ul style="list-style-type: none"> 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	<p>When received and valid:</p> <ul style="list-style-type: none"> The remote prefix SID: <ul style="list-style-type: none"> 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 $N = (\text{SRGB start label of the next-hop IS-IS neighbor}) + \text{SID index}$. 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 N_i, where $N_i = (\text{SRGB start label of next-hop IS-IS neighbor}) + \text{SID index}$.

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 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 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 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 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 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 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 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 true
              hold-time 20
            }
          }
        }
      }
    }
  }
}
```

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 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 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 assign 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 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 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 13001
                }
                ipv6-adjacency-sid {
                  static 14001
                }
              }
            }
          }
        }
      }
    }
  }
}

```

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 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 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 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 Remote LFA with segment routing

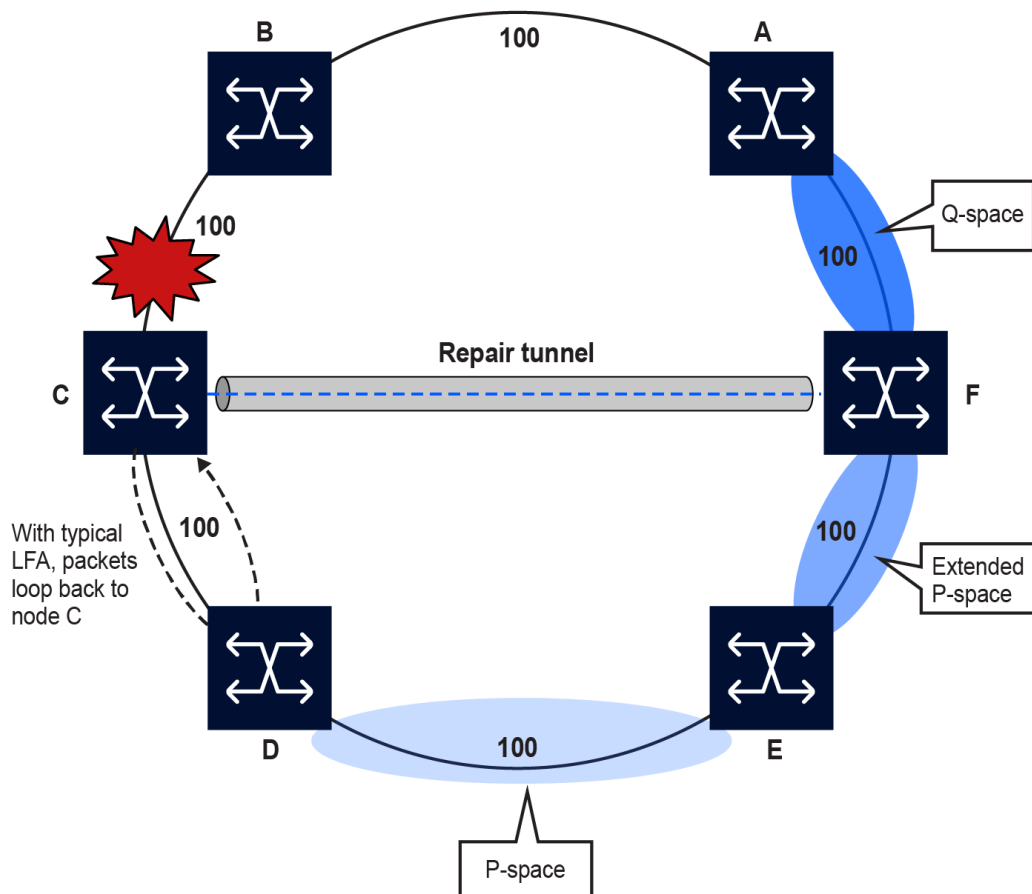
Remote LFA (RLFA) extends the protection coverage of LFA-FRR to any topology by automatically computing and establishing a repair tunnel to a remote LFA node following a link failure. Remote LFA puts the packets back into the shortest path without looping them back to the node that forwarded them.

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.

6.1.1 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 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.

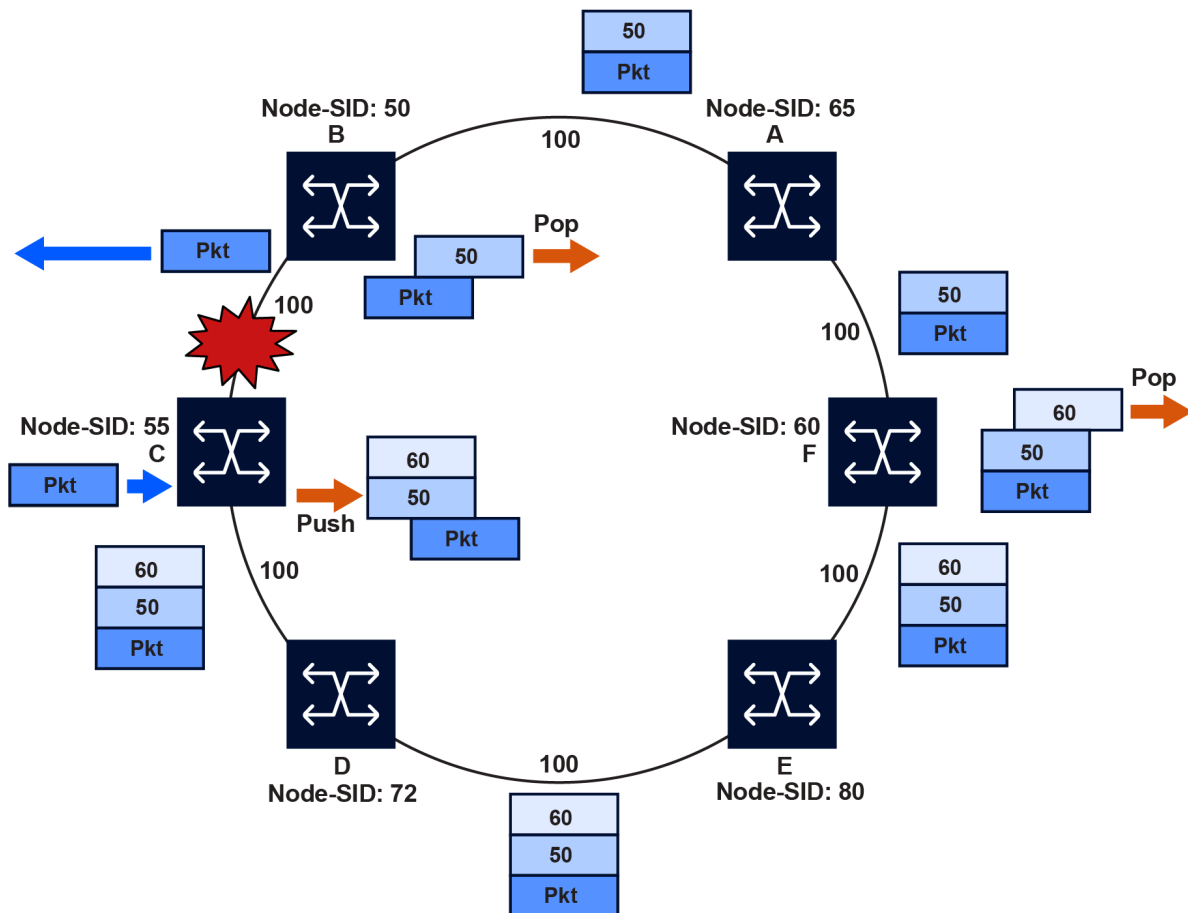
This is the intersection of extended P and Q spaces. The best alternate node (PQ node) is node F in the preceding figure. From node F onwards, traffic follows the IGP shortest path.

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.

6.1.2 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



hw4247

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.

6.1.3 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.

6.1.4 Remote LFA node-protect operation

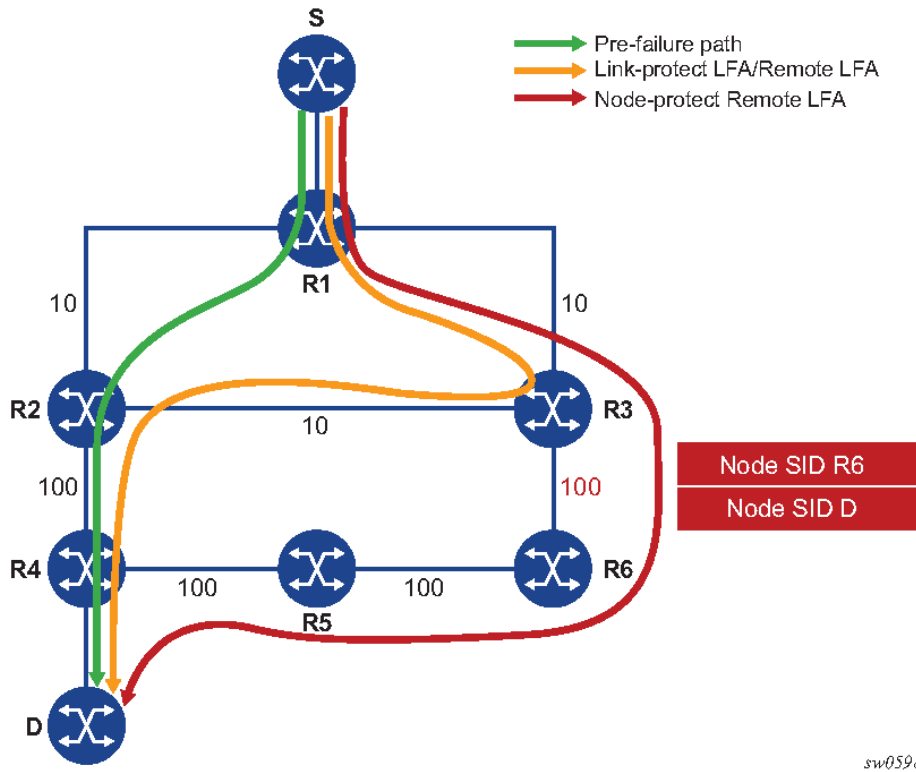
SR Linux supports the node-protect extensions to the remote LFA algorithm, as described in RFC 8102. When node protection is enabled, the router prefers a node-protect over a link-protect repair tunnel in the RLFA SPF computations. This feature protects against the failure of a downstream node. The node-protect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

$$\text{Distance_opt}(R3, Y_i) < \text{Distance_opt}(R3, R2) + \text{Distance_opt}(R2, Y_i)$$

Where $\text{Distance_opt}(A, B)$ 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 Z_i from which node R2 can be reached without any path transiting the protected link R1-R2, using the following equation.

$$\text{Distance_opt}(Z_i, R2) < \text{Distance_opt}(Z_i, R1) + \text{Distance_opt}(R1, R2)$$

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.

$$\text{Distance_opt}(\text{PQ}_i, D) < \text{Distance_opt}(\text{PQ}_i, R2) + \text{Distance_opt}(R2, D)$$

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.

6.2 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 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
                }
            }
        }
    }
}

```

6.2.1 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 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
                    }
                }
            }
        }
    }
}

```

6.2.2 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.

Example: Exclude an IS-IS instance from LFA

```

--{ candidate shared default */--[ ]--
# info network-instance default protocols isis instance 1 level 1 loopfree-alternate-
exclude
    network-instance default {
        protocols {
            isis {
                instance 1 {

```

```

        level 1 {
            loopfree-alternate-exclude true
        }
    }
}

```

6.2.3 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 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
                        }
                    }
                }
            }
        }
    }
}

```

6.3 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:

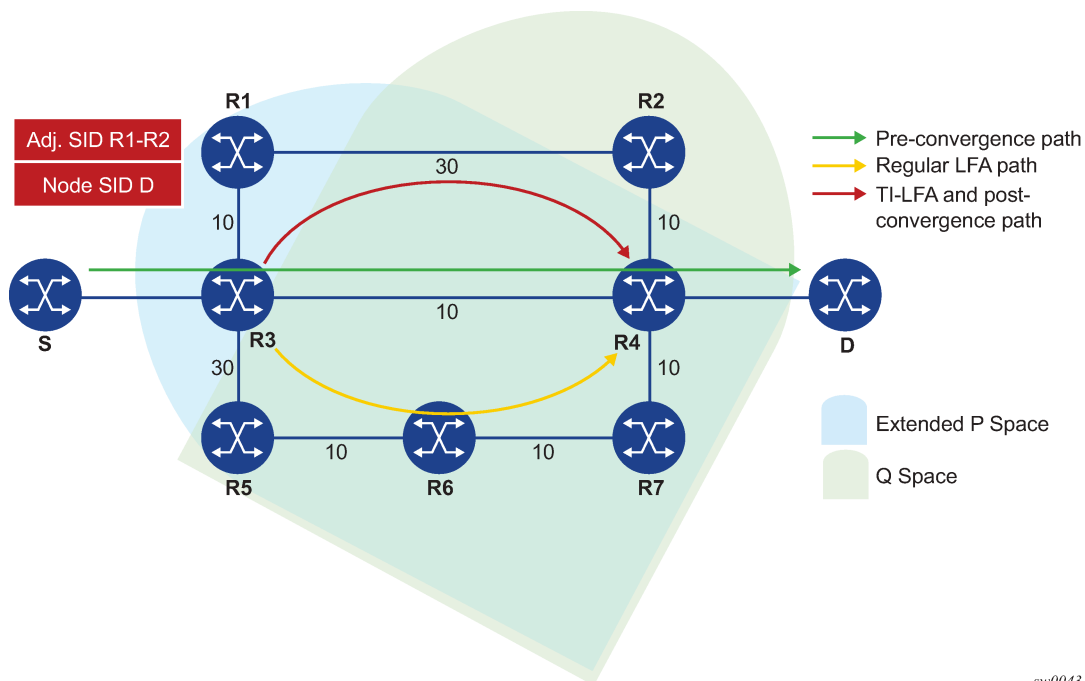
- 7250 IXR
- 7730 SXR

6.3.1 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 Y_i 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:

$$\text{Distance_opt}(R1, Y_i) < \text{Distance_opt}(R1, R3) + \text{Distance_opt}(R3, Y_i)$$

Where $\text{Distance_opt}(A, B)$ 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 Z_i 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.

$$\text{Distance_opt}(Z_i, R4) < \text{Distance_opt}(Z_i, R3) + \text{Distance_opt}(R3, R4)$$

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

6.3.2 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.

6.3.3 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:

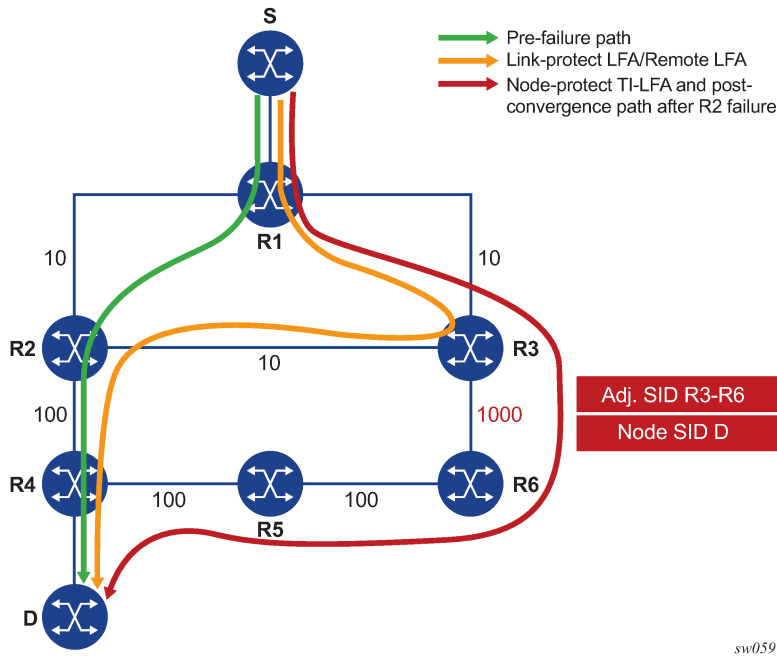
- adjacency SID of an alternate ECMP next hop
- TI-LFA backup next hop
- remote LFA backup next hop

6.3.4 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 Y_i 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 the following calculation to compute the LFA SPF for R3:

$$\text{Distance_opt}(R3, Y_i) < \text{Distance_opt}(R3, R2) + \text{Distance_opt}(R2, Y_i)$$

Where $\text{Distance_opt}(A, B)$ 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 Z_i 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:

$$\text{Distance_opt}(Z_i, R_2) < \text{Distance_opt}(Z_i, R_1) + \text{Distance_opt}(R_1, R_2)$$

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.

6.3.4.1 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.

6.4 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 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 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 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 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 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 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 Type	Tunnel Type	Tunnel ID	FIB	Metric	Preference	Last Update	Next-hop (Type)	Next-hop
10.20.1.1/32	mpls	sr-isis	13010	Y	10	11	2021-11-10T 14:17:32.202Z	10.10.1.1 (mpls)	ethernet-1/33.1
10.20.1.3/32	mpls	sr-isis	13030	Y	10	11	2021-11-10T 14:18:02.299Z	10.10.3.3 (mpls)	ethernet-1/34.1
10.20.1.4/32	mpls	sr-isis	13040	Y	10	11	2021-11-10T 14:18:26.729Z	10.10.4.4 (mpls)	ethernet-1/8.1
10.20.1.5/32	mpls	sr-isis	13050	Y	10	11	2021-11-10T 14:19:02.343Z	10.10.23.5 (mpls)	ethernet-1/9.1
10.20.1.6/32	mpls	sr-isis	13060	Y	10	11	2021-11-10T 14:19:38.140Z	10.10.14.6 (mpls)	ethernet-1/11.1

5 SR-ISIS tunnels, 5 active, 0 inactive

IPv6 tunnel table of network-instance "default"

IPv6 Prefix	Encaps Type	Tunnel Type	Tunnel ID	FIB	Metric	Prefer ence	Last Update	Next-hop (Type)	Next-hop
2001:db8:20: 1::1/128	mpls	sr-isis	13610	Y	10	11	2021-11-10T 14:17:32.204Z	fe80::201: 1ff:feff:20(mpls)	ethernet-1/33.1
2001:db8:20: 1::3/128	mpls	sr-isis	13630	Y	10	11	2021-11-10T 14:18:03.316Z	fe80::201: 3ff:feff:21(mpls)	ethernet-1/34.1
2001:db8:20: 1::4/128	mpls	sr-isis	13640	Y	10	11	2021-11-10T 14:18:31.638Z	fe80::201: 4ff:feff:20(mpls)	ethernet-1/8.1
2001:db8:20: 1::5/128	mpls	sr-isis	13650	Y	10	11	2021-11-10T 14:19:04.882Z	fe80::201: 5ff:feff:20(mpls)	ethernet-1/9.1
2001:db8:20: 1::6/128	mpls	sr-isis	13660	Y	10	11	2021-11-10T 14:19:43.545Z	fe80::201: 6ff:feff:4(mpls)	ethernet-1/11.1

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 Type	Tunnel Type	Tunnel ID	FIB	Metric	Preference	Last Update	Next-hop (Type)	Next-hop
10.20.1.1/32	mpls	sr-isis	13010	Y	10	11	2021-11-10T 14:17:32.202Z	10.10.1.1 (mpls)	ethernet-1/33.1
10.20.1.3/32	mpls	sr-isis	13030	Y	10	11	2021-11-10T 14:18:02.299Z	10.10.3.3 (mpls)	ethernet-1/34.1
10.20.1.4/32	mpls	sr-isis	13040	Y	10	11	2021-11-10T 14:18:26.729Z	10.10.4.4 (mpls)	ethernet-1/8.1
10.20.1.5/32	mpls	sr-isis	13050	Y	10	11	2021-11-10T 14:19:02.343Z	10.10.23.5 (mpls)	ethernet-1/9.1
10.20.1.6/32	mpls	sr-isis	13060	Y	10	11	2021-11-10T	10.10.14.6	ethernet-1/11.1


```

|         |         |         |         |         |         |         |         |         |         |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
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:db8:20:1::1/128|mpls|sr-isis|13610|Y|10|11|2021-11-10T14:17:32.202Z|fe80::201:1ff:feff:20(mpls)|ethernet-1/33.1|
|2001:db8:20:1::3/128|mpls|sr-isis|13630|Y|10|11|2021-11-10T14:18:02.299Z|fe80::201:3ff:feff:21(mpls)|ethernet-1/34.1|
|2001:db8:20:1::4/128|mpls|sr-isis|13640|Y|10|11|2021-11-10T14:18:02.299Z|fe80::201:4ff:feff:20(mpls)|ethernet-1/8.1|
|2001:db8:20:1::5/128|mpls|sr-isis|13650|Y|10|11|2021-11-10T14:18:02.299Z|fe80::201:5ff:feff:20(mpls)|ethernet-1/9.1|
|2001:db8:20:1::6/128|mpls|sr-isis|13660|Y|10|11|2021-11-10T14:18:02.299Z|fe80::201:6ff:feff:4(mpls)|ethernet-1/11.1|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
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 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 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 IGP.

Example: Enabling TE on an interface

```

--{ * candidate shared default }--[ ]--
# info 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 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 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. Links with the same color are grouped into the same class. The admin-group is advertised in sub-TLV (type 3) of the Extended IS Reachability TLV (type 22). Configuring an admin group for TE interface allows network administrators to classify and control routing decisions based on specific criteria or policies.

Example: Configuring admin-groups

In this example, an admin-group named blue is configured.

```
--{ * candidate shared default }--[ ]--
# info 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 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 IGP's 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 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 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 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 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

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 measurements 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 6: 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 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 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 VPRN.

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 (headend) 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 colored and uncolored types.

- 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.
- 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.

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.



Note: Uncolored SR-MPLS TE-policy is supported on 7250 IXR Gen 2, 7250 IXR Gen 2c+, and 7730 SXR platforms.

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 be either strict or loose hop.

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

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 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 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 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 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 [Enabling advertisement of IS-IS TE TLVs/sub-TLVs](#).

Example: Configuring SR-MPLS TE policy

The following example configures the SR-MPLS TE policy using the defined explicit path and the BSID.

```

--{ + candidate shared default }--[ ]--
# info 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 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 to a static MPLS label range

Procedure

You can configure a BSID to reference a static MPLS label range. The high-level steps are as follows:

1. Define the static label range.
2. Assign the static label range to the BSID.

Example: Configure a static MPLS label range

In the configuration example, a static MPLS range (`srl-static-label-range`) is set from 16001-104857. The `start-label` or `end-label` value of the static LSP label range must be within the range 16-1048575.

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

11.1.5 Configuring BSID to a MPLS label value

Procedure

You can configure a BSID to reference a static MPLS label value. The high-level steps are as follows:

1. Configure a static MPLS label range.
2. Assign a static MPLS label value to the BSID

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 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 a static MPLS label value to the BSID

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 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
            }
        }
    }
}
}

```

11.1.6 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.

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) segment list: allows routers within a network to calculate TE Policy segment list paths based on CSPF algorithms autonomously. Unlike explicit paths defined manually, local CSPF computation dynamically selects paths based on constraints and real-time network conditions.
- 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.

11.1.6.1 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.

11.1.6.1.1 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 hop in the path.

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.

Example

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

11.1.6.2 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*

11.1.6.2.1 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 7: 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
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)

11.1.6.2.2 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.

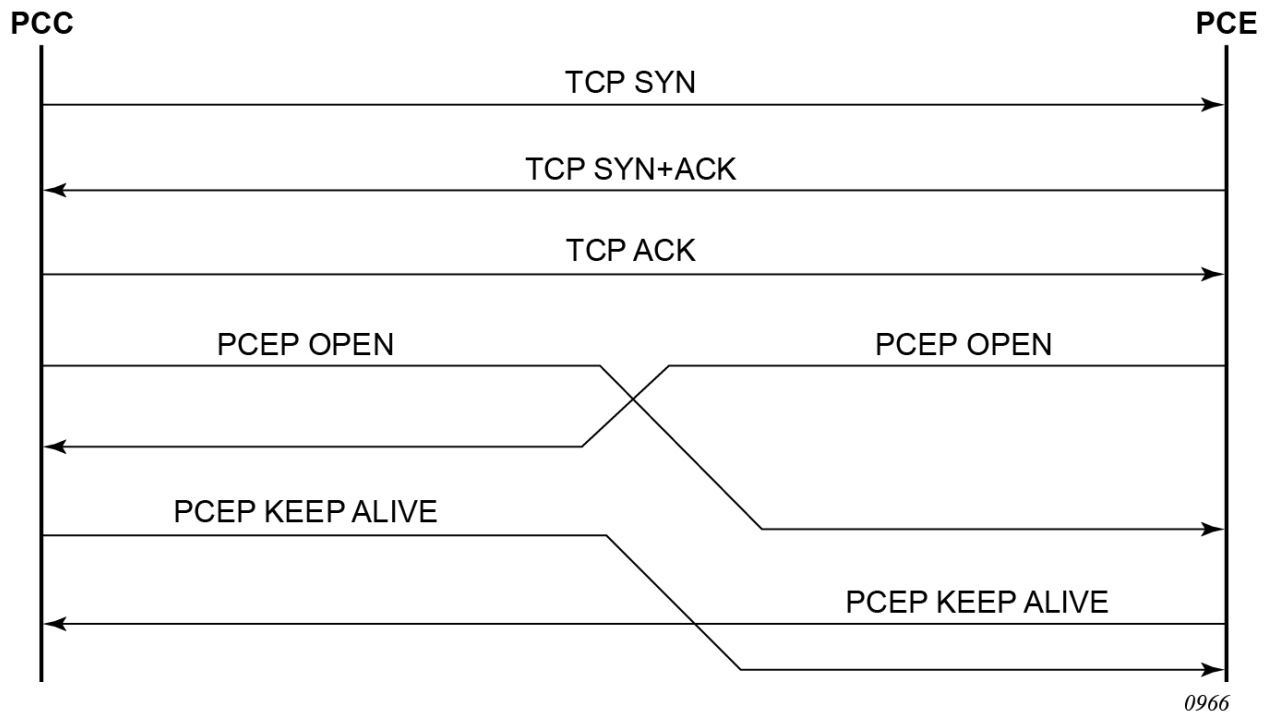
By default, the PCC attempts to reach the remote PCE address through the default network instance. PCE connectivity can be via default or management network instance. SR Linux node in the PCC role attempts an in-band connection to PCE via `default network . instance` which can be changed to `management 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.

11.1.6.2.3 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`
- `relegation-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 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 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 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 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 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 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 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
        }
    }
}

```

11.1.6.2.4 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 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 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
    }
  }
}

```

11.1.6.2.5 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-mp~~ls~~-unco~~l~~ored) 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 8: 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-mp\ls-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-mp\ls-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.

11.1.6.2.6 PCEP establishment and maintenance of TE policy segment list

This section describes the PCEP establishment and maintenance of the TE policy segment lists .

11.1.6.2.6.1 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.

11.1.6.2.6.1.1 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.

11.1.6.2.6.1.1.1 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 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
        }
      }
    }
  }
}
```

11.1.6.2.6.1.2 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?`.

11.1.6.2.6.1.2.1 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 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 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
          }
        }
      }
    }
  }
}

```

1.1.6.2.6.1.2.2 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 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
        }
      }
    }
  }
}

```

11.1.6.2.6.2 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:

- Generic association object with Association ID and Association type, defined in RFC 8697
- 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

11.1.6.2.6.2.1 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, PCRep. 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 9: PCEP ASSOCIATION object and message

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

The following are the PCEP extensions for IPv4 LSPs:

Table 10: PCEP numbers (IPv4 LSPs)

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

Table 11: 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 12: PCEP numbers

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

Table 13: Path Protection Association TLV

Type	Name	Reference
1	Path Protection Association	RFC 8745
38	Path Protection Association Group TLV	

Table 14: Path Protection Association TLV Flag

Bit	Name	Reference
31	P - PROTECTION-LSP	RFC 8745
30	S - SECONDARY-LSP	
6-29	Unassigned	

Bit	Name	Reference
0-5	Protection Type Flags	

11.1.6.2.6.2.2 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 15: PCEP TLVs

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

Table 16: 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 17: NO-PATH-VECTOR bit flags

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

11.1.6.2.6.2.2.1 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 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
            }
          }
        }
      }
    }
  }
}
```

11.1.6.2.6.2.3 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 18: ASSOCIATION type field

Value	Name	Reference
3	Policy Association	RFC 9005

Table 19: PCEP TLV type indicators

Value	Meaning	Reference
48	POLICY-PARAMETERS-TLV	RFC 9005

1.1.6.2.6.2.3.1 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 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
            }
          }
        }
      }
    }
  }
}
```

11.1.6.2.6.2.4 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 20: 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 21: 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 22: 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 23: 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

11.1.6.3 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.

11.1.6.3.1 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 SID protection when LFA, RLFA, or TI-LFA is enabled. It is possible to remove SID protection on a specific link by setting the parameter `local-sr-protection` within the `network-instance traffic-engineering-policies policy segment-list dynamic path-algorithm local-cspf 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.

11.1.6.3.2 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

```
--{ +* candidate shared default }--[ ]--
info network-instance default traffic-engineering-policies policy srl_test_policy
  network-instance default {
    traffic-engineering-policies {
      policy sr-mpls-te-policy {
        policy-type sr-mpls-uncolored
        endpoint 10.0.0.2
        tag-set srl-tag-set
        segment-list 3 {
          dynamic {
            path-algorithm local-cspf
          }
        }
        segment-list 4 {
          pce-control true
        }
      }
    }
  }
}
```

```

    }
  }
}

```

11.1.6.3.3 Local CSPF path computation using delay metric

A TE-policy segment list used by real-time and delay-sensitive user and control traffic can have its path computed by local CSPF based on the 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 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 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 total delay of the calculated path must stay within this limit.

The TE-policy segment list is only resigned 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.

11.1.6.3.3.1 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.

```

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

11.1.6.4 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 of PCEP session failure or when the redelegation 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 24: 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 methodfallback-path-algorithm	PCE state	Behaviour
pce	local-cspf	Not configured	PCE is operational	Upon primary path falure, 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

Primary path algorithm (path-algorithm)	Secondary path algorithm (path-algorithm)	Fallback path algorithm method fallback-path-algorithm	PCE state	Behaviour
				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 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.

11.1.6.4.1 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 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
          }
        }
      }
    }
  }
```

11.1.7 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, VPRN, 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.

11.1.7.1 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 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 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
      }
    }
  }
```

11.1.7.2 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 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
          }
        }
      }
    }
  }
```

```
    }
  }
}
```

11.1.7.3 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 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
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

11.1.8 Uncolored SR-MPLS TE path failure codes

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

Table 25: 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
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

Failure codes	Description
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
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

11.2 Colored TE policy



Note: Colored TE policies are supported on 7250 IXR Gen 2, 7250 IXR Gen 2c+, and 7730 SXR platforms; However, the protection mode feature is currently not supported for colored TE policies

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.

**Note:**

Weighted ECMP across segment lists of a given candidate path is not qualified despite any valid weight value programmed. Similarly, datapath ECMP resources are allocated in line with the weights.

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 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 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 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 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 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 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.



Note:

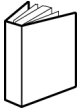
Weighted ECMP across segment list within a candidate path is not supported, even if valid weight values are programmed. The datapath ECMP resources are allocated according to the specified weights.

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.

Customer document and product support



Customer documentation

[Customer documentation welcome page](#)



Technical support

[Product support portal](#)



Documentation feedback

[Customer documentation feedback](#)