# Segment Routing Configuration Guide for Cisco ASR 9000 Series Routers, IOS XR Release 6.2.x

**First Published:** 2017-03-17

**Americas Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

# CONTENTS

# Preface

**Note** This product has reached end-of-life status. For more information, see the End-of-Life and End-of-Sale Notices.

From Release 6.1.2 onwards, Cisco introduces support for the 64-bit Linux-based IOS XR operating system. Extensive feature parity is maintained between the 32-bit and 64-bit environments. Unless explicitly marked otherwise, the contents of this document are applicable for both the environments. For more details on Cisco IOS XR 64 bit, refer to the Release Notes for Cisco ASR 9000 Series Routers, Release 6.1.2 document.

The *Segment Routing Configuration Guide for Cisco ASR 9000 Series Aggregation Services Routers* preface contains these sections:



## Changes to This Document

This table lists the changes made to this document since it was first printed.

| Date | Change Summary |
|---|---|
| March 2017 | Initial release of this document |

## Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

### Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

CHAPTER 1

# New and Changed Information for Segment Routing Features

This table summarizes the new and changed feature information for the *Segment Routing Configuration Guide for Cisco ASR 9000 Aggregation Services Routers* , and lists where they are documented.



# New and Changed Information

***Table 1: New and Changed Features***

| Feature | Description | Introduced/Changed in Release | Where Documented |
|---|---|---|---|
| IOS XR Traffic Controller (XTC) | This feature was introduced. | Release 6.2.1 | *Configure IOS XR Traffic Controller (XTC)* |
| Segment Routing Microloop Avoidance | This feature was introduced. | Release 6.2.1 | *Configure Segment Routing Microloop Avoidance* |
| IS-IS Multi-Domain Prefix SID and Domain Stitching | This feature was introduced. | Release 6.2.1 | *Configure Segment Routing for IS-IS Protocol* |
| Strict SPF Segment IDs for IS-IS | This feature was introduced. | Release 6.2.1 | *Configure Segment Routing for IS-IS Protocol* |
| Prefix Attribute TLV for IS-IS | This feature was introduced. | Release 6.2.1 | *Configure Segment Routing for IS-IS Protocol* |
| MPLS Ping and Traceroute for Prefix-SID | This feature was introduced. | Release 6.2.1 | *Using Segment Routing OAM* |

CHAPTER 2

# About Segment Routing

**Note** Segment Routing is not supported on 1st generation Cisco ASR 9000 Ethernet Line Cards or the Cisco ASR 9000 SIP-700 SPA Interface Processor. Refer to the Cisco ASR 9000 Ethernet Line Card Installation Guide for details about 1st generation line cards.

This chapter introduces the concept of segment routing and provides a workflow for configuring segment routing.



## Scope

Segment routing is a method of forwarding packets on the network based on the source routing paradigm. The source chooses a path and encodes it in the packet header as an ordered list of segments. Segments are an identifier for any type of instruction. For example, topology segments identify the next hop toward a destination. Each segment is identified by the segment ID (SID) consisting of a flat unsigned 20-bit integer.

### Segments

Interior gateway protocol (IGP) distributes two types of segments: prefix segments and adjacency segments. Each router (node) and each link (adjacency) has an associated segment identifier (SID).

- A prefix SID is associated with an IP prefix. The prefix SID is manually configured from the segment routing global block (SRGB) range of labels, and is distributed by IS-IS or OSPF. The prefix segment steers the traffic along the shortest path to its destination. A node SID is a special type of prefix SID that identifies a specific node. It is configured under the loopback interface with the loopback address of the node as the prefix.

  A prefix segment is a global segment, so a prefix SID is globally unique within the segment routing domain.

- An adjacency segment is identified by a label called an adjacency SID, which represents a specific adjacency, such as egress interface, to a neighboring router. The adjacency SID is distributed by IS-IS or OSPF. The adjacency segment steers the traffic to a specific adjacency.

An adjacency segment is a local segment, so the adjacency SID is locally unique relative to a specific router.

By combining prefix (node) and adjacency segment IDs in an ordered list, any path within a network can be constructed. At each hop, the top segment is used to identify the next hop. Segments are stacked in order at the top of the packet header. When the top segment contains the identity of another node, the receiving node uses equal cost multipaths (ECMP) to move the packet to the next hop. When the identity is that of the receiving node, the node pops the top segment and performs the task required by the next segment.

### Dataplane

Segment routing can be directly applied to the Multiprotocol Label Switching (MPLS) architecture with no change in the forwarding plane. A segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels. The segment to process is on the top of the stack. The related label is popped from the stack, after the completion of a segment.

### Services

Segment Routing integrates with the rich multi-service capabilities of MPLS, including Layer 3 VPN (L3VPN), Virtual Private Wire Service (VPWS), Virtual Private LAN Service (VPLS), and Ethernet VPN (EVPN).

### Segment Routing for Traffic Engineering

Segment routing for traffic engineering (SR-TE) takes place through a tunnel between a source and destination pair. Segment routing for traffic engineering uses the concept of source routing, where the source calculates the path and encodes it in the packet header as a segment. Each segment is an end-to-end path from the source to the destination, and instructs the routers in the provider core network to follow the specified path instead of the shortest path calculated by the IGP. The destination is unaware of the presence of the tunnel.

# Need

With segment routing for traffic engineering (SR-TE), the network no longer needs to maintain a per-application and per-flow state. Instead, it simply obeys the forwarding instructions provided in the packet.

SR-TE utilizes network bandwidth more effectively than traditional MPLS-TE networks by using ECMP at every segment level. It uses a single intelligent source and relieves remaining routers from the task of calculating the required path through the network.

# Benefits

- **Ready for SDN**: Segment routing was built for SDN and is the foundation for Application Engineered Routing (AER). SR prepares networks for business models, where applications can direct network behavior. SR provides the right balance between distributed intelligence and centralized optimization and programming.
- **Minimal configuration**: Segment routing for TE requires minimal configuration on the source router.
- **Load balancing**: Unlike in RSVP-TE, load balancing for segment routing can take place in the presence of equal cost multiple paths (ECMPs).

- **Supports Fast Reroute (FRR)**: Fast reroute enables the activation of a pre-configured backup path within 50 milliseconds of path failure.
- **Plug-and-Play deployment**: Segment routing tunnels are interoperable with existing MPLS control and data planes and can be implemented in an existing deployment.

# Workflow for Deploying Segment Routing

Follow this workflow to deploy segment routing.

1. Configure the Segment Routing Global Block (SRGB)
2. Enable Segment Routing and Node SID for the IGP
3. Configure Segment Routing for BGP
4. Configure the SR-TE Policy
5. Configure TI-LFA and Microloop Avoidance
6. Configure the Segment Routing Mapping Server
7. Collect Traffic Statistics

CHAPTER 3

# Configure Segment Routing Global Block

Local label allocation is managed by the label switching database (LSD). The Segment Routing Global Block (SRGB) is the range of label values preserved for segment routing in the LSD.



# About the Segment Routing Global Block

The SRGB label values are assigned as prefix segment identifiers (SIDs) to SR-enabled nodes and have global significance throughout the domain.

**Note** Because the values assigned from the range have domain-wide significance, we recommend that all routers within the domain be configured with the same range of values.

The default SRGB range is from 16000 to 23999.

**Note** On SR-capable routers, the default starting value of the dynamic label range is increased from 16000 to 24000, so that the default SRGB label values (16000 to 23999) are available when SR is enabled on a running system. If a dynamic label range has been configured with a starting value of 16000, then the default SRGB label values may already be in use when SR is enabled on a running system. Therefore, you must reload the router after enabling SR to release the currently allocated labels and allocate the SRGB.

Also, if you need to increase the SRGB range after you have enabled SR, you must reload the router to release the currently allocated labels and allocate the new SRGB.

To keep the segment routing configuration simple and to make it easier to troubleshoot segment routing issues, we recommend that you use the default SRGB range on each node in the domain. However, there are instances when you might need to define a different range. For example:

- The nodes of another vendor support a label range that is different from the default SRGB, and you want to use the same SRGB on all nodes.
- The default range is too small.
- To specify separate SRGBs for IS-IS and OSPF protocols, as long as the ranges do not overlap.

**Restrictions:**

- In Cisco IOS XR release 6.2.x and earlier, LSD label values 0-15999 are reserved.
- In Cisco IOS XR release 6.2.x and earlier, the maximum SRGB size is 65536.
- The SRGB upper bound cannot exceed the platform's capability.

**Note** Label values that are not previously reserved are available for dynamic assignment.

The SRGB can be disabled if SR is not used.

# Setup a Non-Default Segment Routing Global Block Range

This task explains how to configure a non-default SRGB range.

### SUMMARY STEPS

1. **configure**
2. [**router** {**isis** *instance-id* | **ospf** *process_name*} ]
3. **segment-routing global-block** *starting_value ending_value*
4. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | [**router** {**isis** *instance-id* \| **ospf** *process_name*} ]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router isis 1` | (Optional) Enter the **router isis** *instance-id* or **router ospf** *process_name* commands if you want to configure separate SRGBs for IS-IS and OSPF protocols. |
| **Step 3** | **segment-routing global-block** *starting_value ending_value*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis)# segment-routing global-block 18000 19999` | Enter the lowest value that you want the SRGB range to include as the starting value. Enter the highest value that you want the SRGB range to include as the ending value. |
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions: |

| | Command or Action | Purpose |
|---|---|---|
| | | • **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify the SRGB configuration:

```
RP/0/RSP0/CPU0:router# show mpls label table detail
Table Label   Owner                           State  Rewrite
----- ------- ------------------------------- ------ -------

<...snip...>

0     18000   ISIS(A):1                       InUse  No
   Lbl-blk SRGB, vers:0, (start_label=18000, size=2000)
0     24000   ISIS(A):1                       InUse  Yes
   (SR Adj Segment IPv4, vers:0, index=1, type=0, intf=Gi0/0/0/0, nh=10.0.0.2)
```

### What to do next

Configure prefix SIDs and enable segment routing.

CHAPTER 4

# Configure Segment Routing for IS-IS Protocol

Integrated Intermediate System-to-Intermediate System (IS-IS), Internet Protocol Version 4 (IPv4), is a standards-based Interior Gateway Protocol (IGP). The Cisco IOS XR software implements the IP routing capabilities described in International Organization for Standardization (ISO)/International Engineering Consortium (IEC) 10589 and RFC 1995, and adds the standard extensions for single topology and multitopology IS-IS for IP Version 6 (IPv6).

This module provides the configuration information used to enable segment routing for IS-IS.

**Note** For additional information on implementing IS-IS on your Cisco ASR 9000 Series Router, see the *Implementing IS-IS* module in the *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*.



# Enabling Segment Routing for IS-IS Protocol

Segment routing on the IS-IS control plane supports the following:

- IPv4 and IPv6 control plane
- Level 1, level 2, and multi-level routing
- Prefix SIDs for host prefixes on loopback interfaces
- Multiple IS-IS instances on the same loopback interface for domain border nodes
- Adjacency SIDs for adjacencies
- MPLS penultimate hop popping (PHP) and explicit-null signaling

This task explains how to enable segment routing for IS-IS.

### Before you begin

Your network must support the MPLS Cisco IOS XR software feature before you enable segment routing for IS-IS on your router.

**Note** You must enter the commands in the following task list on every IS-IS router in the traffic-engineered portion of your network.

### SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family** { **ipv4** | **ipv6** } [ **unicast** ]
4. **metric-style wide** [ **level** { **1** | **2** }]
5. **mpls traffic-eng** *level*
6. **mpls traffic-eng router-id** *interface*
7. **router-id loopback** *loopback interface used for prefix-sid*
8. **segment-routing mpls**
9. **exit**
10. **mpls traffic-eng**
11. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router isis** *instance-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router isis isp` | Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.<br>**Note** You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command. |
| **Step 3** | **address-family** { **ipv4** \| **ipv6** } [ **unicast** ]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast` | Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. |
| **Step 4** | **metric-style wide** [ **level** { **1** \| **2** }]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# metric-style wide level 1` | Configures a router to generate and accept only wide link metrics in the Level 1 area. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 5** | **mpls traffic-eng** *level*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng level-2-only` | Enables RSVP traffic engineering funtionality. |
| **Step 6** | **mpls traffic-eng router-id** *interface*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng router-id Loopback0` | Sets the traffic engineering loopback interface. |
| **Step 7** | **router-id loopback** *loopback interface used for prefix-sid*<br>**Example:**<br>`RP/0/(config-isis-af)#router-id loopback0` | Configures router ID for each address-family (ipv4/ipv6). |
| **Step 8** | **segment-routing mpls**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# segment-routing mpls` | Segment routing is enabled by the following actions:<br>• MPLS forwarding is enabled on all interfaces where IS-IS is active.<br>• All known prefix-SIDs in the forwarding plain are programmed, with the prefix-SIDs advertised by remote routers or learned through local or remote mapping server.<br>• The prefix-SIDs locally configured are advertised. |
| **Step 9** | **exit**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# exit`<br>`RP/0/RSP0/CPU0:router(config-isis)# exit` | |
| **Step 10** | **mpls traffic-eng**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# mpls traffic-eng` | Enables traffic engineering functionality on the node. The node advertises the traffic engineering link attributes in IGP which populates the traffic engineering database (TED) on the head-end. The RSVP-TE head-end requires the TED to calculate and validate the path of the RSVP-TE policy. |
| **Step 11** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes. |

| | Command or Action | Purpose |
|---|---|---|
| | | • **Cancel** —Remains in the configuration session, without committing the configuration changes. |

### What to do next

Configure the prefix SID.

# Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface

A prefix segment identifier (SID) is associated with an IP prefix. The prefix SID is manually configured from the segment routing global block (SRGB) range of labels. A prefix SID is configured under the loopback interface with the loopback address of the node as the prefix. The prefix segment steers the traffic along the shortest path to its destination.

A prefix SID can be a node SID or an Anycast SID. A node SID is a type of prefix SID that identifies a specific node. An Anycast SID is a type of prefix SID that identifies a set of nodes, and is configured with n-flag clear. The set of nodes (Anycast group) is configured to advertise a shared prefix address and prefix SID. Anycast routing enables the steering of traffic toward multiple advertising nodes. Packets addressed to an Anycast address are forwarded to the topologically nearest nodes.

Strict-SPF SIDs are used to forward traffic strictly along the SPF path. Strict-SPF SIDs are not forwarded to SR-TE tunnels. IS-IS advertises the SR Algorithm sub Type Length Value (TLV) (in the SR Router Capability SubTLV) to include both algorithm 0 (SPF) and algorithm 1 (Strict-SPF). When the IS-IS area or level is Strict-SPF TE-capable, Strict-SPF SIDs are used to build the SR-TE Strict-SPF tunnels. Strict-SPF SIDs are also used to program the backup paths for prefixes, node SIDs, and adjacency SIDs.

**Note** The same SRGB is used for both regular SIDs and strict-SPF SIDs.

The prefix SID is globally unique within the segment routing domain.

This task explains how to configure prefix segment identifier (SID) index or absolute value on the IS-IS enabled Loopback interface.

### Before you begin

Ensure that segment routing is enabled on the corresponding address family.

### SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface Loopback** *instance*
4. **address-family** { **ipv4** | **ipv6** } [ **unicast** ]
5. **prefix-sid** [**strict-spf** ] {**index** *SID-index* | **absolute** *SID-value*} [**n-flag-clear**] [**explicit-null** ]
6. Use the **commit** or **end** command.

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| Step 1 | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| Step 2 | **router isis** *instance-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)#` **`router isis 1`** | Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.<br>• You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command. |
| Step 3 | **interface Loopback** *instance*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis)#` **`interface Loopback0`** | Specifies the loopback interface and instance. |
| Step 4 | **address-family** { **ipv4** \| **ipv6** } [ **unicast** ]<br>**Example:**<br>The following is an example for ipv4 address family:<br>`RP/0/RSP0/CPU0:router(config-isis-if)#` **`address-family ipv4 unicast`** | Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. |
| Step 5 | **prefix-sid** [**strict-spf** ] {**index** *SID-index* \| **absolute** *SID-value*} [**n-flag-clear**] [**explicit-null** ]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-if-af)#` **`prefix-sid index 1001`**<br>`RP/0/RSP0/CPU0:router(config-isis-if-af)#` **`prefix-sid strict-spf index 101`**<br>`RP/0/RSP0/CPU0:router(config-isis-if-af)#` **`prefix-sid absolute 17001`** | Configures the prefix-SID index or absolute value for the interface.<br>Specify **strict-spf** to configure the prefix-SID to use the SPF path instead of the SR-TE tunnel.<br>Specify **index** *SID-index* for each node to create a prefix SID based on the lower boundary of the SRGB + the index.<br>Specify **absolute** *SID-value* for each node to create a specific prefix SID within the SRGB.<br>By default, the n-flag is set on the prefix-SID, indicating that it is a node SID. For specific prefix-SID (for example, Anycast prefix-SID), enter the `n-flag-clear` keyword. IS-IS does not set the `N` flag in the prefix-SID sub Type Length Value (TLV).<br>To disable penultimate-hop-popping (PHP) and add explicit-Null label, enter `explicit-null` keyword. IS-IS sets the `E` flag in the prefix-SID sub TLV. |

| | Command or Action | Purpose |
|---|---|---|
| | | **Note** IS-IS does not advertise separate explicit-NULL or flags for regular SIDs and strict-SPF SIDs. The settings in the regular SID are used if the settings are different. |
| **Step 6** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify the prefix-SID configuration:

```
RP/0/RSP0/CPU0:router# show isis database verbose

IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
router.00-00        * 0x0000039b   0xfc27        1079             0/0/0
  Area Address: 49.0001
  NLPID:        0xcc
  NLPID:        0x8e
  MT:           Standard (IPv4 Unicast)
  MT:           IPv6 Unicast                                      0/0/0
  Hostname:     router
  IP Address:   10.0.0.1
  IPv6 Address: 2001:0db8:1234::0a00:0001
  Router Cap:   10.0.0.1, D:0, S:0
    Segment Routing: I:1 V:1, SRGB Base: 16000 Range: 8000
    SR Algorithm:
      Algorithm: 0
      Algorithm: 1
<...>
  Metric: 0          IP-Extended 10.0.0.1/32
    Prefix-SID Index: 1001, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix-SID Index: 101, Algorithm:1, R:0 N:1 P:0 E:0 V:0 L:0
<...>
```

### What to do next

Configure the SR-TE policy.

# IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability

The following sub-TLVs support the advertisement of IPv4 and IPv6 prefix attribute flags and the source router ID of the router that originated a prefix advertisement, as described in RFC 7794.

- Prefix Attribute Flags
- IPv4 and IPv6 Source Router ID

## Prefix Attribute Flags

The Prefix Attribute Flag sub-TLV supports the advertisement of attribute flags associated with prefix advertisements. Knowing if an advertised prefix is directly connected to the advertising router helps to determine how labels that are associated with an incoming packet should be processed.

This section describes the behavior of each flag when a prefix advertisement is learned from one level to another.

**Note** Prefix attributes are only added when wide metric is used.

### Prefix Attribute Flags Sub-TLV Format

```
 0 1 2 3 4 5 6 7 ...
+-+-+-+-+-+-+-+-+...
|X|R|N|          ...
+-+-+-+-+-+-+-+-+...
```

### Prefix Attribute Flags Sub-TLV Fields

| Field | Description |
|---|---|
| X (External Prefix Flag) | This flag is set if the prefix has been redistributed from another protocol. The value of the flag is preserved when the prefix is propagated to another level. |
| R (Re-advertisement Flag) | This flag is set to 1 by the Level 1-2 router when the prefix is propagated between IS-IS levels (from Level 1 to Level 2, or from Level 2 to Level 1).<br>This flag is set to 0 when the prefix is connected locally to an IS-IS-enabled interface (regardless of the level configured on the interface). |

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>N (Node Flag)</td><td>For prefixes that are propagated from another level:<br>1. Copy the N-flag from the prefix attribute sub-TLV, if present in the source level.<br>2. Copy the N-flag from the prefix-SID sub-TLV, if present in the source level.<br>3. Otherwise, set to 0.<br><br>For connected prefixes:<br>1. Set to 0 if prefix-attributes n-flag-clear is configured (see Configuring Prefix Attribute N-flag-clear).<br>2. Set to 0 if n-flag-clear{ n-flag-clearSID-index | n-flag-clearSID-value} n-flag-clear is configured (see Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface).<br>3. Otherwise, set to 1 when the prefix is a host prefix (/32 for IPV4, /128 for IPv6) that is associated with a loopback address.<br>Note If the flag is set and the prefix length is not a host prefix, then the flag must be ignored.</td></tr>
</table>

# IPv4 and IPv6 Source Router ID

The Source Router ID sub-TLV identifies the source of the prefix advertisement. The IPv4 and IPv6 source router ID is displayed in the output of the **show isis database verbose** command.

The Source Router ID sub-TLV is added when the following conditions are met:

1. The prefix is locally connected.
2. The N-flag is set to 1 (when it's a host prefix and the **n-flag-clear** configuration is not used).
3. The router ID is configured in the corresponding address family.

The source router ID is propagated between levels.

***Table 2: Source Router Sub-TLV Format***

| | |
|---|---|
| IPv4 Source Router ID | Type: 11<br>Length: 4<br>Value: IPv4 Router ID of the source of the prefix advertisement |
| IPv6 Source Router ID | Type: 12<br>Length: 16<br>Value: IPv6 Router ID of the source of the prefix advertisement |

# Configuring Prefix Attribute N-flag-clear

The N-flag is set to 1 when the prefix is a host prefix (/32 for IPV4, /128 for IPv6) that is associated with a loopback address. The advertising router can be configured to not set this flag. This task explains how to clear the N-flag.

## SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface Loopback** *instance*
4. **prefix-attributes n-flag-clear** [**Level-1** | **Level-2**]
5. Use the **commit** or **end** command.

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router isis** *instance-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router isis 1` | |
| **Step 3** | **interface Loopback** *instance*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface Loopback0` | Specifies the loopback interface. |
| **Step 4** | **prefix-attributes n-flag-clear** [**Level-1** \| **Level-2**]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# isis prefix-attributes n-flag-clear` | Clears the prefix attribute N-flag explicitly. |
| **Step 5** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes. |

| | Command or Action | Purpose |
|---|---|---|
| | | • **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify the prefix attribute configuration:

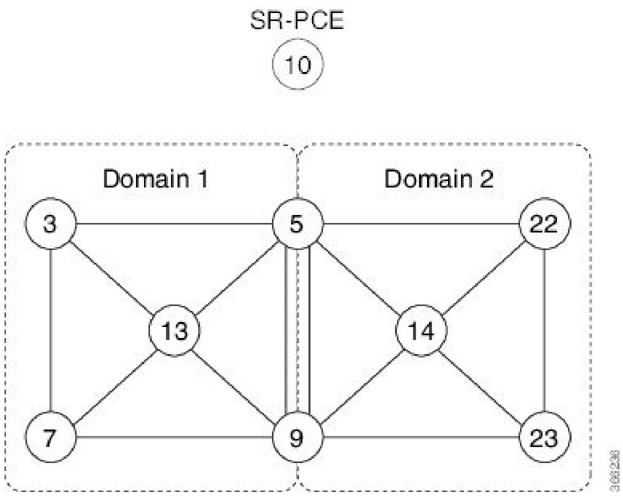
```
RP/0/RSP0/CPU0:router# show isis database verbose

IS-IS 1 (Level-2) Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
router.00-00        * 0x0000039b   0xfc27        1079             0/0/0
  Area Address: 49.0001
  NLPID:        0xcc
  NLPID:        0x8e
  MT:           Standard (IPv4 Unicast)
  MT:           IPv6 Unicast                                      0/0/0
  Hostname:     router
  IP Address:   10.0.0.1
  IPv6 Address: 2001:0db8:1234::0a00:0001
  Router Cap:   10.0.0.1, D:0, S:0
    Segment Routing: I:1 V:1, SRGB Base: 16000 Range: 8000
    SR Algorithm:
      Algorithm: 0
      Algorithm: 1
<...>
  Metric: 0          IP-Extended 10.0.0.1/32
    Prefix-SID Index: 1001, Algorithm:0, R:1 N:0 P:1 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:1 N:0
  Metric: 10         IP-Extended 10.0.0.2/32
    Prefix-SID Index: 1002, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
    Prefix Attribute Flags: X:0 R:0 N:1
    Source Router ID: 10.0.0.2
<...>
```

# IS-IS Multi-Domain Prefix SID and Domain Stitching: Example

IS-IS Multi-Domain Prefix SID and Domain Stitching allows you to configure multiple IS-IS instances on the same loopback interface for domain border nodes. You specify a loopback interface and prefix SID under multiple IS-IS instances to make the prefix and prefix SID reachable in different domains.

This example uses the following topology. Node 5 and 9 are border nodes between two IS-IS domains (Domain1 and Domain2). Node 10 is configured as the Segment Routing Path Computation Element (SR-PCE) (see Configure Segment Routing Path Computation Element).

*Figure 1: Multi-Domain Topology*




# Configure IS-IS Multi-Domain Prefix SID

Specify a loopback interface and prefix SID under multiple IS-IS instances on each border node:

```
Example: Border Node 5
router isis Domain1
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16005

router isis Domain2
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16005
```

```
Example: Border Node 9
router isis Domain1
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16009

router isis Domain2
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid absolute 16009
```

Border nodes 5 and 9 each run two IS-IS instances (Domain1 and Domain2) and advertise their Loopback0 prefix and prefix SID in both domains.

Nodes in both domains can reach the border nodes by using the same prefix and prefix SID. For example, Node 3 and Node 22 can reach Node 5 using prefix SID 16005.

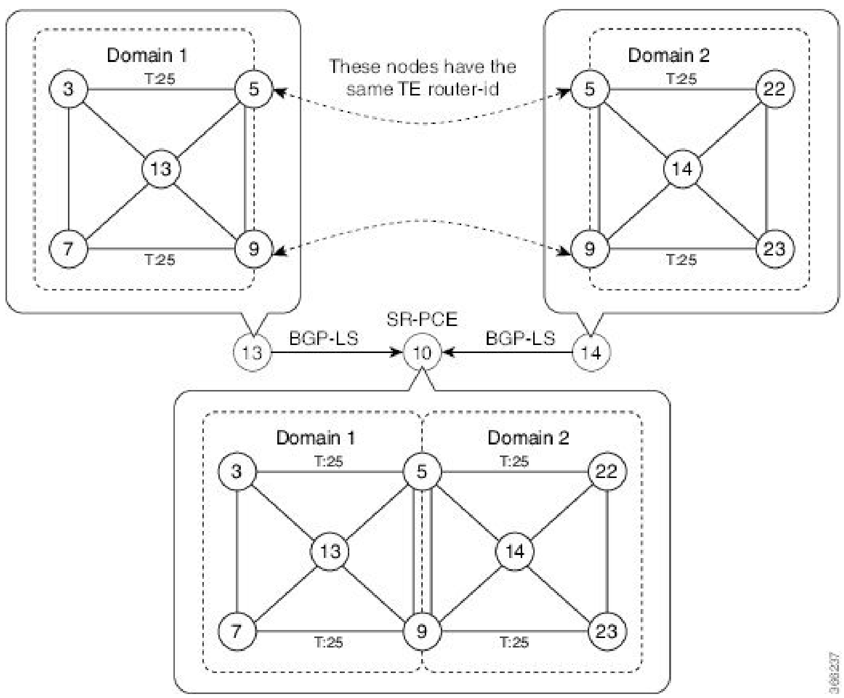
# Configure Common Router ID

On each border node, configure a common TE router ID under each IS-IS instance:

```
Example: Border Node 5
router isis Domain1
 address-family ipv4 unicast
  router-id loopback0

router isis Domain2
 address-family ipv4 unicast
  router-id loopback0
```

```
Example: Border Node 9
router isis Domain1
 address-family ipv4 unicast
  router-id loopback0

router isis Domain2
 address-family ipv4 unicast
  router-id loopback0
```

# Distribute IS-IS Link-State Data




Configure BGP Link-state (BGP-LS) on Node 13 and Node 14 to report their local domain to Node 10:

```
Example: Node 13
router isis Domain1
 distribute link-state instance-id instance-id
```

```
Example: Node 14
router isis Domain2
 distribute link-state instance-id instance-id
```

Link-state ID starts from 32. One ID is required per IGP domain. Different domain IDs are essential to identify that the SR-TE TED belongs to a particular IGP domain.

Nodes 13 and 14 each reports its local domain in BGP-LS to Node 10.

Node 10 identifies the border nodes (Nodes 5 and 9) by their common advertised TE router ID, then combines (stitches) the domains on these border nodes for end-to-end path computations.

CHAPTER 5

# Configure Segment Routing for OSPF Protocol

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets.

This module provides the configuration information to enable segment routing for OSPF.

**Note** For additional information on implementing OSPF on your Cisco ASR 9000 Series Router, see the *Implementing OSPF* module in the *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*.



## Enabling Segment Routing for OSPF Protocol

Segment routing on the OSPF control plane supports the following:

- OSPFv2 control plane
- Multi-area
- IPv4 prefix SIDs for host prefixes on loopback interfaces
- Adjacency SIDs for adjacencies
- MPLS penultimate hop popping (PHP) and explicit-null signaling

This section describes how to enable segment routing MPLS and MPLS forwarding in OSPF. Segment routing can be configured at the instance, area, or interface level.

### Before you begin

Your network must support the MPLS Cisco IOS XR software feature before you enable segment routing for OSPF on your router.

**Note** You must enter the commands in the following task list on every OSPF router in the traffic-engineered portion of your network.

### SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **mpls traffic-eng router-id** *interface*
4. **segment-routing mpls**
5. **area** *area*
6. **mpls traffic-eng**
7. **segment-routing mpls**
8. **exit**
9. **mpls traffic-eng**
10. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router ospf** *process-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router ospf 1` | Enables OSPF routing for the specified routing process and places the router in router configuration mode. |
| **Step 3** | **mpls traffic-eng router-id** *interface*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)# mpls traffic-eng router-id Loopback0` | Sets the traffic engineering loopback interface. |
| **Step 4** | **segment-routing mpls**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)# segment-routing mpls` | Enables segment routing using the MPLS data plane on the routing process and all areas and interfaces in the routing process.<br>Enables segment routing fowarding on all interfaces in the routing process and installs the SIDs received by OSPF in the forwarding table. |
| **Step 5** | **area** *area*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)# area 0` | Enters area configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 6** | **mpls traffic-eng**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# mpls traffic-eng` | Enables IGP traffic engineering funtionality. |
| **Step 7** | **segment-routing mpls**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# segment-routing mpls` | (Optional) Enables segment routing using the MPLS data plane on the area and all interfaces in the area. Enables segment routing fowarding on all interfaces in the area and installs the SIDs received by OSPF in the forwarding table. |
| **Step 8** | **exit**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# exit`<br>`RP/0/RSP0/CPU0:router(config-ospf)# exit` | |
| **Step 9** | **mpls traffic-eng**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# mpls traffic-eng` | Enables traffic engineering funtionality on the node. The node advertises the traffic engineering link attributes in IGP which populates the traffic engineering database (TED) on the head-end. The SR-TE head-end requires the TED to calculate and validate the path of the SR-TE policy. |
| **Step 10** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

### What to do next

Configure the prefix SID.

# Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface

A prefix segment identifier (SID) is associated with an IP prefix. The prefix SID is manually configured from the segment routing global block (SRGB) range of labels. A prefix SID is configured under the loopback interface with the loopback address of the node as the prefix. The prefix segment steers the traffic along the shortest path to its destination.

A prefix SID can be a node SID or an Anycast SID. A node SID is a type of prefix SID that identifies a specific node. An Anycast SID is a type of prefix SID that identifies a set of nodes, and is configured with n-flag clear. The set of nodes (Anycast group) is configured to advertise a shared prefix address and prefix SID. Anycast routing enables the steering of traffic toward multiple advertising nodes. Packets addressed to an Anycast address are forwarded to the topologically nearest nodes.

The prefix SID is globally unique within the segment routing domain.

This task describes how to configure prefix segment identifier (SID) index or absolute value on the OSPF-enabled Loopback interface.

### Before you begin

Ensure that segment routing is enabled on an instance, area, or interface.

## SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **area** *value*
4. **interface Loopback** *interface-instance*
5. **prefix-sid** [**strict-spf**] {**index** *SID-index* | **absolute** *SID-value* } [**n-flag-clear**] [**explicit-null**]
6. Use the **commit** or **end** command.

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router ospf** *process-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router ospf 1` | Enables OSPF routing for the specified routing process, and places the router in router configuration mode. |
| **Step 3** | **area** *value*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)# area 0` | Enters area configuration mode. |
| **Step 4** | **interface Loopback** *interface-instance*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Loopback0 passive` | Specifies the loopback interface and instance. |
| **Step 5** | **prefix-sid** [**strict-spf**] {**index** *SID-index* \| **absolute** *SID-value* } [**n-flag-clear**] [**explicit-null**] | Configures the prefix-SID index or absolute value for the interface. |

| | Command or Action | Purpose |
|---|---|---|
| | **Example:**<br><br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# prefix-sid index 1001`<br><br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# prefix-sid absolute 17001` | Specify **strict-spf** to configure the prefix-SID to use the SPF path instead of the SR-TE tunnel.<br><br>Specify **index** *SID-index* for each node to create a prefix SID based on the lower boundary of the SRGB + the index.<br><br>Specify **absolute** *SID-value* for each node to create a specific prefix SID within the SRGB.<br><br>By default, the n-flag is set on the prefix-SID, indicating that it is a node SID. For specific prefix-SID (for example, Anycast prefix-SID), enter the `n-flag-clear` keyword. OSPF does not set the `N` flag in the prefix-SID sub Type Length Value (TLV).<br><br>To disable penultimate-hop-popping (PHP) and add an explicit-Null label, enter the `explicit-null` keyword. OSPF sets the `E` flag in the prefix-SID sub TLV. |
| **Step 6** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br><br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify the prefix-SID configuration:

```
RP/0/RSP0/CPU0:router# show ospf database opaque-area 7.0.0.1 self-originate
 OSPF Router with ID (10.0.0.1) (Process ID 1)
               Type-10 Opaque Link Area Link States (Area 0)
<...>
    Extended Prefix TLV: Length: 20
      Route-type: 1
      AF        : 0
      Flags     : 0x40
      Prefix    : 10.0.0.1/32

      SID sub-TLV: Length: 8
        Flags     : 0x0
        MTID      : 0
        Algo      : 0
        SID Index : 1001
```

### What to do next

Configure SR-TE Policies

CHAPTER 6

# Configure Segment Routing for BGP

Border Gateway Protocol (BGP) is an Exterior Gateway Protocol (EGP) that allows you to create loop-free inter-domain routing between autonomous systems. An autonomous system is a set of routers under a single technical administration. Routers in an autonomous system can use multiple Interior Gateway Protocols (IGPs) to exchange routing information inside the autonomous system and an EGP to route packets outside the autonomous system.

This module provides the configuration information used to enable Segment Routing for BGP.

**Note** For additional information on implementing BGP on your Cisco ASR 9000 Series Router, see the *Implementing BGP* module in the *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*.



## Segment Routing for BGP

In a traditional BGP-based data center (DC) fabric, packets are forwarded hop-by-hop to each node in the autonomous system. Traffic is directed only along the external BGP (eBGP) multipath ECMP. No traffic engineering is possible.

In an MPLS-based DC fabric, the eBGP sessions between the nodes exchange BGP labeled unicast (BGP-LU) network layer reachability information (NLRI). An MPLS-based DC fabric allows any leaf (top-of-rack or border router) in the fabric to communicate with any other leaf using a single label, which results in higher packet forwarding performance and lower encapsulation overhead than traditional BGP-based DC fabric. However, since each label value might be different for each hop, an MPLS-based DC fabric is more difficult to troubleshoot and more complex to configure.

BGP has been extended to carry segment routing prefix-SID index. BGP-LU helps each node learn BGP prefix SIDs of other leaf nodes and can use ECMP between source and destination. Segment routing for BGP simplifies the configuration, operation, and troubleshooting of the fabric. With segment routing for BGP, you can enable traffic steering capabilities in the data center using a BGP prefix SID.

# Configure BGP Prefix Segment Identifiers

Segments associated with a BGP prefix are known as BGP prefix SIDs. The BGP prefix SID is global within a segment routing or BGP domain. It identifies an instruction to forward the packet over the ECMP-aware best-path computed by BGP to the related prefix. The BGP prefix SID is manually configured from the segment routing global block (SRGB) range of labels.

Each BGP speaker must be configured with an SRGB using the **segment-routing global-block** command. See the About the Segment Routing Global Block section for information about the SRGB.

**Note** Because the values assigned from the range have domain-wide significance, we recommend that all routers within the domain be configured with the same range of values.

To assign a BGP prefix SID, first create a routing policy using the **set label-index** *index* attribute, then associate the index to the node.

**Note** A routing policy with the **set label-index** attribute can be attached to a network configuration or redistribute configuration. Other routing policy language (RPL) configurations are possible. For more information on routing policies, refer to the "Implementing Routing Policy" chapter in the *Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide*.

### Example

The following example shows how to configure the SRGB, create a BGP route policy using a $SID parameter and **set label-index** attribute, and then associate the prefix-SID index to the node.

```
RP/0/RSP0/CPU0:router(config)# segment-routing global-block 16000 23999

RP/0/RSP0/CPU0:router(config)# route-policy SID($SID)
RP/0/RSP0/CPU0:router(config-rpl)# set label-index $SID
RP/0/RSP0/CPU0:router(config-rpl)# end policy

RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)# network 1.1.1.3/32 route-policy SID(3)
RP/0/RSP0/CPU0:router(config-bgp-af)# allocate-label all
RP/0/RSP0/CPU0:router(config-bgp-af)# commit
RP/0/RSP0/CPU0:router(config-bgp-af)# end


RP/0/RSP0/CPU0:router# show bgp 1.1.1.3/32
BGP routing table entry for 1.1.1.3/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 74          74
    Local Label: 16003
Last Modified: Sep 29 19:52:18.155 for 00:07:22
Paths: (1 available, best #1)
  Advertised to update-groups (with more than one peer):
    0.2
```

```
Path #1: Received by speaker 0
Advertised to update-groups (with more than one peer):
  0.2
3
  99.3.21.3 from 99.3.21.3 (1.1.1.3)
    Received Label 3
    Origin IGP, metric 0, localpref 100, valid, external, best, group-best
    Received Path ID 0, Local Path ID 1, version 74
    Origin-AS validity: not-found
    Label Index: 3
```

# Segment Routing Egress Peer Engineering

Segment routing egress peer engineering (EPE) uses a controller to instruct an ingress provider edge, or a content source (node) within the segment routing domain, to use a specific egress provider edge (node) and a specific external interface to reach a destination. BGP peer SIDs are used to express source-routed inter-domain paths.

Below are the BGP-EPE peering SID types:

- PeerNode SID—To an eBGP peer. Pops the label and forwards the traffic on any interface to the peer.
- PeerAdjacency SID—To an eBGP peer via interface. Pops the label and forwards the traffic on the related interface.

The controller learns the BGP peer SIDs and the external topology of the egress border router through BGP-LS EPE routes. The controller can program an ingress node to steer traffic to a destination through the egress node and peer node using BGP labeled unicast (BGP-LU).

EPE functionality is only required at the EPE egress border router and the EPE controller.

## Configure Segment Routing Egress Peer Engineering

This task explains how to configure segment routing EPE on the EPE egress node.

### SUMMARY STEPS

1. **router bgp** *as-number*
2. **neighbor** *ip-address*
3. **remote-as** *as-number*
4. **egress-engineering**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **router bgp** *as-number*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router bgp 1` | Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **neighbor** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.1.3` | Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer. |
| **Step 3** | **remote-as** *as-number*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 3` | Creates a neighbor and assigns a remote autonomous system number to it. |
| **Step 4** | **egress-engineering**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr)# egress-engineering` | Configures the egress node with EPE for the eBGP peer. |

# Configure BGP Link-State

BGP Link-State (LS) is an Address Family Identifier (AFI) and Sub-address Family Identifier (SAFI) defined to carry interior gateway protocol (IGP) link-state database through BGP. BGP LS delivers network topology information to topology servers and Application Layer Traffic Optimization (ALTO) servers. BGP LS allows policy-based control to aggregation, information-hiding, and abstraction. BGP LS supports IS-IS and OSPFv2.

**Note** IGPs do not use BGP LS data from remote peers. BGP does not download the received BGP LS data to any other component on the router.

For segment routing, the following attributes have been added to BGP LS:

- Node—Segment routing capability (including SRGB range) and algorithm
- Link—Adjacency SID and LAN adjacency SID
- Prefix—Prefix SID and segment routing mapping server (SRMS) prefix range

The following example shows how to exchange link-state information with a BGP neighbor:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router bgp 1
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family link-state link-state
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# exit
```

### IGP Link-State Database Distribution

A given BGP node may have connections to multiple, independent routing domains. IGP link-state database distribution into BGP-LS is supported for both OSPF and IS-IS protocols in order to distribute this information on to controllers or applications that desire to build paths spanning or including these multiple domains.

To distribute IS-IS link-state data using BGP LS, use the **distribute link-state** command in router configuration mode.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router isis isp
RP/0/RSP0/CPU0:router(config-isis)# distribute link-state instance-id 32 level 2 throttle
5
```

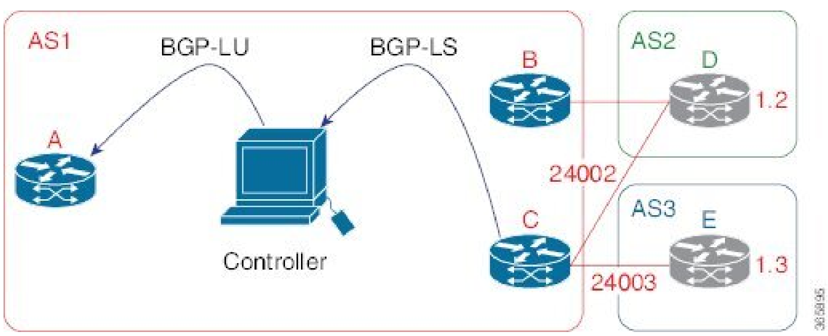
To distribute OSPFv2 link-state data using BGP LS, use the **distribute link-state** command in router configuration mode.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# router ospf 100
RP/0/RSP0/CPU0:router(config-ospf)# distribute link-state instance-id 32 throttle 10
```

# Example: Configuring SR-EPE and BGP-LS

In the following figure, segment routing is enabled on autonomous system AS1 with ingress node A and egress nodes B and C. In this example, we configure EPE on egress node C.

*Figure 2: Topology*




**Step 1** Configure node C with EPE for eBGP peers D and E.

**Example:**

```
RP/0/RSP0/CPU0:router_C(config)# router bgp 1
RP/0/RSP0/CPU0:router_C(config-bgp)# neighbor 192.168.1.3
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# remote-as 3
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# description to E
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# egress-engineering
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_in in
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_out out
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# exit
```

```
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# exit
RP/0/RSP0/CPU0:router_C(config-bgp)# neighbor 192.168.1.2
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# remote-as 2
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# description to D
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# egress-engineering
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_in in
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# route-policy bgp_out out
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# exit
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# exit
```

**Step 2** Configure node C to advertise peer node SIDs to the controller using BGP-LS.

**Example:**

```
RP/0/RSP0/CPU0:router_C(config-bgp)# neighbor 172.29.50.71
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# remote-as 1
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# description to EPE_controller
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# address-family link-state link-state
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# exit
RP/0/RSP0/CPU0:router_C(config-bgp)# exit
```

**Step 3** Commit the configuration.

**Example:**

```
RP/0/RSP0/CPU0:router_C(config)# commit
```

**Step 4** Verify the configuration.

**Example:**

```
RP/0/RSP0/CPU0:router_C# show bgp egress-engineering

 Egress Engineering Peer Set: 192.168.1.2/32 (10b87210)
     Nexthop: 192.168.1.2
     Version: 2, rn_version: 2
       Flags: 0x00000002
   Local ASN: 1
  Remote ASN: 2
   Local RID: 1.1.1.3
  Remote RID: 1.1.1.4
   First Hop: 192.168.1.2
        NHID: 3
       Label: 24002, Refcount: 3
     rpc_set: 10b9d408

 Egress Engineering Peer Set: 192.168.1.3/32 (10be61d4)
     Nexthop: 192.168.1.3
     Version: 3, rn_version: 3
       Flags: 0x00000002
   Local ASN: 1
  Remote ASN: 3
   Local RID: 1.1.1.3
  Remote RID: 1.1.1.5
   First Hop: 192.168.1.3
        NHID: 4
       Label: 24003, Refcount: 3
     rpc_set: 10be6250
```

The output shows that node C has allocated peer SIDs for each eBGP peer.

### Example:

```
RP/0/RSP0/CPU0:router_C# show mpls forwarding labels 24002 24003
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
24002  Unlabelled  No ID              Te0/3/0/0    192.168.1.2     0
24003  Unlabelled  No ID              Te0/1/0/0    192.168.1.3     0
```

The output shows that node C installed peer node SIDs in the Forwarding Information Base (FIB).

CHAPTER 7

# Configure SR-TE Policies

This module provides information about segment routing for traffic engineering (SR-TE) policies, how to configure SR-TE policies, and how to steer traffic into an SR-TE policy.



# About SR-TE Policies

Segment routing for traffic engineering (SR-TE) uses a "policy" to steer traffic through the network. An SR-TE policy path is expressed as a list of segments that specifies the path, called a segment ID (SID) list. Each segment is an end-to-end path from the source to the destination, and instructs the routers in the network to follow the specified path instead of the shortest path calculated by the IGP. If a packet is steered into an SR-TE policy, the SID list is pushed on the packet by the head-end. The rest of the network executes the instructions embedded in the SID list.

There are two types of SR-TE policies: dynamic and explicit.

### Local Dynamic SR-TE Policy

When you configure local dynamic SR-TE, the head-end locally calculates the path to the destination address. Dynamic path calculation results in a list of interface IP addresses that traffic engineering (TE) maps to adj-SID labels. Routes are learned by way of forwarding adjacencies over the TE tunnel.

### Explicit SR-TE Policy

An explicit path is a list of IP addresses or labels, each representing a node or link in the explicit path. This feature is enabled through the **explicit-path** command that allows you to create an explicit path and enter a configuration submode for specifying the path.

# How to Configure SR-TE Policies

This section contains the following procedures:

# Configure Local Dynamic SR-TE Policy

This task explains how to configure a local dynamic SR-TE policy.

### SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **ipv4 unnumbered** *type interface-path-id*
4. **destination** *ip-address*
5. **path-option** *preference-priority* **dynamic segment-routing**
6. **path-protection**
7. **commit**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface tunnel-te** *tunnel-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface tunnel-te22` | Configures the tunnel interface. |
| **Step 3** | **ipv4 unnumbered** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0` | Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type. |
| **Step 4** | **destination** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2` | Assigns a destination address on the new tunnel. |
| **Step 5** | **path-option** *preference-priority* **dynamic segment-routing**<br>**Example:** | Sets the path option to dynamic and assigns the path ID. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/RSP0/CPU0:router(config-if)# path-option 1 dynamic segment-routing` | |
| **Step 6** | **path-protection**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# path-protection` | Enables path protection on the tunnel-te interface. |
| **Step 7** | **commit** | |

This completes the configuration of the dynamic SR-TE policy.

# Configure Explicit SR-TE Policy

This task explains how to configure an explicit SR-TE policy.

### SUMMARY STEPS

1. **configure**
2. **explicit-path name** *path-name*
3. **index** *index* {**next-address** *ip-address* | **next-label** *label*}
4. **exit**
5. **interface tunnel-te** *tunnel-id*
6. **ipv4 unnumbered** *type interface-path-id*
7. **destination** *ip-address* [**verbatim**]
8. **path-option** *preference-priority* **explicit name** *path-name* **segment-routing**
9. **commit**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **explicit-path name** *path-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# explicit-path name r1r6_exp` | Enters a name for the explicit path and enters the explicit path configuration mode. |
| **Step 3** | **index** *index* {**next-address** *ip-address* \| **next-label** *label*}<br>**Example:** | Specifies a label or an address in an explicit path of a tunnel. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/RSP0/CPU0:router(config-expl-path)# index 1 next-label 16001`<br>`RP/0/RSP0/CPU0:router(config-expl-path)# index 2 next-label 16006` | **Note** • You can include multiple addresses, labels, or both. However, you cannot configure addresses after you have configured labels. Once you start configuring labels, you need to continue with labels.<br>• Each entry must have a unique index.<br>• If the first hop is specified as next-label, that label must be an Adj-SID of the head-end or a prefix-SID label value known by the head-end. |
| **Step 4** | **exit** | |
| **Step 5** | **interface tunnel-te** *tunnel-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface tunnel-te22` | Configures the tunnel interface. |
| **Step 6** | **ipv4 unnumbered** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0` | Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type. |
| **Step 7** | **destination** *ip-address* [**verbatim**]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2` | Assigns a destination address on the new tunnel.<br>Typically, the tunnel destination must have a match in the routing information base (RIB). For inter-area or inter-domain policies to destinations that are otherwise not reachable, use the **verbatim** option to disable the RIB verification on a tunnel destination. |
| **Step 8** | **path-option** *preference-priority* **explicit name** *path-name* **segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# path-option 1 explicit name r1r6_exp segment-routing` | Specifies the explicit path name and assigns the path ID. |
| **Step 9** | **commit** | |

This completes the configuration of the explicit SR-TE policy.

# Steering Traffic into an SR-TE Policy

This section describes the following traffic steering methods:

### Static Routes

Static routes can use the segment routing tunnel as a next-hop interface. Both IPv4 and IPv6 prefixes can be routed through the tunnel.

A static route to a destination with a prefix-SID removes the IGP-installed SR-forwarding entry of that prefix.

### Autoroute Announce

The SR-TE policy can be advertised into an IGP as a next hop by configuring the autoroute announce statement on the source router. The IGP then installs routes in the Routing Information Base (RIB) for shortest paths that involve the tunnel destination. Autoroute announcement of IPv4 prefixes can be carried through either OSPF or IS-IS. Autoroute announcement of IPv6 prefixes can be carried only through IS-IS.

### Autoroute Destination

Autoroute destination allows you to automatically route traffic through a segment routing tunnel instead of manually configuring static routes. Multiple autoroute destination addresses can be added in the routing information base (RIB) per tunnel.

Static routes are always added with zero cost metric, which can result in traffic that is mapped on multiple tunnels to always load-balance due to ECMP. This load-balancing may be undesirable when some of those tunnels have sub-optimal paths. With autoroute destination, only the tunnel whose IGP cost to its endpoint is lowest will be considered for carrying traffic.

- **Interaction Between Static Routes and Autoroute Destination**

  If there is a manually configured static route to the same destination as a tunnel with autoroute destination enabled, traffic for that destination is load-shared between the static route and the tunnel with autoroute destination enabled.

- **Interaction Between Autoroute Announce and Autoroute Destination**

  For intra-area tunnels, if a tunnel is configured with both autoroute announce and autoroute destination, the tunnel is announced to the RIB by both the IGP and the static process. RIBs prefer static routes, not IGP routes, so the autoroute destination features takes precedence over autoroute announce.

# Configure Static Routes

This task explains how to configure a static route.

### SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **ipv4 unnumbered** *type interface-path-id*
4. **destination** *ip-address*
5. **path-option** *preference-priority* **dynamic segment-routing**

6. **exit**
7. **router static**
8. **address-family ipv4 unicast**
9. *prefix mask interface-type interface-instance*
10. **commit**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface tunnel-te** *tunnel-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface tunnel-te22` | Configures the tunnel interface. |
| **Step 3** | **ipv4 unnumbered** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0` | Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type. |
| **Step 4** | **destination** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2` | Assigns a destination address on the new tunnel. |
| **Step 5** | **path-option** *preference-priority* **dynamic segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# path-option 1 dynamic segment-routing` | Sets the path option to dynamic and assigns the path ID. |
| **Step 6** | **exit** | |
| **Step 7** | **router static**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router static` | Configures the static route and enters static configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 8** | **address-family ipv4 unicast**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast` | Enters address family mode. |
| **Step 9** | *prefix mask interface-type interface-instance*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-static-af)# 192.168.0.2/32 tunnel-te22` | Specifies the destination prefix is directly reachable through the tunnel interface. |
| **Step 10** | **commit** | |

This completes the configuration of the static route.

# Configure Autoroute Announce

This task explains how to configure autoroute announce to steer traffic through the SR-TE policy.

### SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **ipv4 unnumbered** *type interface-path-id*
4. **autoroute announce**
5. **destination** *ip-address*
6. **path-option** *preference-priority* **dynamic segment-routing**
7. **path-protection**
8. **commit**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface tunnel-te** *tunnel-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface tunnel-te22` | Configures the tunnel interface. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **ipv4 unnumbered** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0` | Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type. |
| **Step 4** | **autoroute announce**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# autoroute announce` | Enables messages that notify the neighbor nodes about the routes that are forwarding. |
| **Step 5** | **destination** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2` | Assigns a destination address on the new tunnel. |
| **Step 6** | **path-option** *preference-priority* **dynamic segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# path-option 1 dynamic segment-routing` | Sets the path option to dynamic and assigns the path ID. |
| **Step 7** | **path-protection**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# path-protection` | Enables path protection on the tunnel-te interface. |
| **Step 8** | **commit** | |

# Configure Autoroute Destination

This task explains how to configure autoroute destination to steer traffic through the SR-TE policy.

### SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **ipv4 unnumbered** *type interface-path-id*
4. **autoroute destination** *destination-ip-address*
5. **destination** *ip-address*
6. **path-option** *preference-priority* **dynamic segment-routing**
7. **commit**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| Step 1 | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| Step 2 | **interface tunnel-te** *tunnel-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# interface tunnel-te22` | Configures the tunnel interface. |
| Step 3 | **ipv4 unnumbered** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0` | Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type. |
| Step 4 | **autoroute destination** *destination-ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.1`<br>`RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.2  (the default route)`<br>`RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.3`<br>`RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.4` | (Optional) Adds a route (destination-ip-address) in the RIB with the tunnel as outgoing interface to the tunnel destination. |
| Step 5 | **destination** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2` | Assigns a destination address on the new tunnel. |
| Step 6 | **path-option** *preference-priority* **dynamic segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-if)# path-option 1 dynamic segment-routing` | Sets the path option to dynamic and assigns the path ID. |
| Step 7 | **commit** | |

# BGP SR-TE

SR-TE can be used by data center (DC) operators to provide different levels of Service Level Assurance (SLA). Setting up SR-TE paths using BGP (BGP SR-TE) simplifies DC network operation without introducing a new protocol for this purpose.

### Explicit BGP SR-TE

Explicit BGP SR-TE uses an SR-TE policy (identified by a unique color ID) that contains a list of explicit paths with SIDs that correspond to each explicit path. A BGP speaker signals an explicit SR-TE policy to a remote peer, which triggers the setup of a TE tunnel with specific characteristics and explicit paths. On the receiver side, a TE tunnel that corresponds to the explicit path is setup by BGP. The packets for the destination mentioned in the BGP update follow the explicit path described by the policy. Each policy can include multiple explicit paths, and TE will create a tunnel for each path.

IPv4 and IPv6 SR policies can be advertised over BGPv4 or BGPv6 sessions between the SR-TE controller and the SR-TE headend. The Cisco IOS-XR implementation supports the following combinations:

- IPv4 SR policy advertised over BGPv4 session
- IPv6 SR policy advertised over BGPv4 session
- IPv6 SR policy advertised over BGPv6 session

## Configure Dynamic BGP SR-TE

Perform this task to configure dynamic BGP SR-TE with low latency per community (optimized TE metric):

### Before you begin

The following configuration must be applied on the head-end router:

```
Router(config)# ipv4 unnumbered mpls traffic-eng Loopback0
Router(config)# mpls traffic-eng
Router(config-mpls-te)# auto-tunnel p2p tunnel-id min number max number
```

### SUMMARY STEPS

1. **configure**
2. **route-policy** *route-policy-name*
3. **end-policy**
4. **commit**
5. **configure**
6. **mpls traffic-eng**
7. **attribute-set p2p-te** *attribute-set-name*
8. **path-selection metric te**
9. **commit**
10. **configure**

11. **router bgp** *as-number*
12. **neighbor** *ip-address*
13. **address-family** {**ipv4** | **ipv6**} **unicast**
14. **route-policy** *route-policy-name* {**in** | **out**
15. **commit**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| Step 1 | **configure** | |
| Step 2 | **route-policy** *route-policy-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# route-policy BGP_TE`<br>`RP/0/RSP0/CPU0:router(config-rpl)# if community matches-every (100:1) then`<br>`RP/0/RSP0/CPU0:router(config-rpl-if)# set mpls traffic-eng attribute-set low_lat`<br>`RP/0/RSP0/CPU0:router(config-rpl-if)# endif`<br>`RP/0/RSP0/CPU0:router(config-rpl)# pass` | Creates a route policy and enters route policy configuration mode, where you can define the route policy. |
| Step 3 | **end-policy**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-rpl)# end-policy` | Ends the definition of a route policy and exits route policy configuration mode. |
| Step 4 | **commit** | |
| Step 5 | **configure** | |
| Step 6 | **mpls traffic-eng**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# mpls traffic-eng` | Enters the MPLS-TE submode. |
| Step 7 | **attribute-set p2p-te** *attribute-set-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(configmpls-te)# attribute-set p2p-te low_lat` | Specifies the name of the point-to-point traffic-engineered (P2P TE) attribute-set and enters attribute-set configuration mode. |
| Step 8 | **path-selection metric te**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-te-attribute-set)# path-selection metric te` | Sets the path-selection metric to use the TE metric. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 9** | **commit** | |
| **Step 10** | **configure** | |
| **Step 11** | **router bgp** *as-number*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router bgp 1` | Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process. |
| **Step 12** | **neighbor** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.1.5` | Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer. |
| **Step 13** | **address-family** {**ipv4** \| **ipv6**} **unicast**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast` | Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. |
| **Step 14** | **route-policy** *route-policy-name* {**in** \| **out**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy BGP_TE in` | Applies the specified policy to inbound IPv4 unicast routes. |
| **Step 15** | **commit** | |

# Configure Explicit BGP SR-TE

Perform this task to configure explicit BGP SR-TE:

### SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **bgp router-id** *ip-address*
4. **address-family** {**ipv4** | **ipv6**} **sr-policy**
5. **exit**
6. **neighbor** *ip-address*
7. **remote-as** *as-number*
8. **address-family** {**ipv4** | **ipv6**} **sr-policy**
9. **route-policy** *route-policy-name* {**in** | **out**}

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure** | |
| **Step 2** | **router bgp** *as-number*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router bgp 65000` | Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process. |
| **Step 3** | **bgp router-id** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 1.1.1.1` | Configures the local router with a specified router ID. |
| **Step 4** | **address-family** {**ipv4** \| **ipv6**} **sr-policy**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 sr-policy` | Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. |
| **Step 5** | **exit** | |
| **Step 6** | **neighbor** *ip-address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.0.1` | Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer. |
| **Step 7** | **remote-as** *as-number*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1` | Creates a neighbor and assigns a remote autonomous system number to it. |
| **Step 8** | **address-family** {**ipv4** \| **ipv6**} **sr-policy**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 sr-policy` | Specifies either the IPv4 or IPv6 address family and enters address family configuration submode. |
| **Step 9** | **route-policy** *route-policy-name* {**in** \| **out**}<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#` | Applies the specified policy to IPv4 or IPv6 unicast routes. |

| | Command or Action | Purpose |
|---|---|---|
| | **route-policy pass out** | |

### Example: BGP SR-TE with BGPv4 Neighbor to BGP SR-TE Controller

The following configuration shows the an SR-TE head-end with a BGPv4 session towards a BGP SR-TE controller. This BGP session is used to signal both IPv4 and IPv6 SR policies.

```
router bgp 65000
bgp router-id 1.1.1.1
 !
 address-family ipv4 sr-policy
 !
 address-family ipv6 sr-policy
 !
 neighbor 10.1.3.1
  remote-as 10
  description *** eBGP session to BGP SRTE controller ***
  address-family ipv4 sr-policy
   route-policy pass in
   route-policy pass out
  !
  address-family ipv6 sr-policy
   route-policy pass in
   route-policy pass out
  !
 !
!
```

### Example: BGP SR-TE with BGPv6 Neighbor to BGP SR-TE Controller

The following configuration shows an SR-TE head-end with a BGPv6 session towards a BGP SR-TE controller. This BGP session is used to signal IPv6 SR policies.

```
router bgp 65000
bgp router-id 1.1.1.1
 address-family ipv6 sr-policy
 !
 neighbor 3001::10:1:3:1
  remote-as 10
  description *** eBGP session to BGP SRTE controller ***
  address-family ipv6 sr-policy
   route-policy pass in
   route-policy pass out
  !
 !
!
```

# Using Binding Segments

The binding segment is a local segment identifying an SR-TE policy. Each SR-TE policy is associated with a binding segment ID (BSID). The BSID is a local label that is automatically allocated for each SR-TE policy when the SR-TE policy is instantiated.

BSID can be used to steer traffic into the SR-TE policy and across domain borders, creating seamless end-to-end inter-domain SR-TE policies. Each domain controls its local SR-TE policies; local SR-TE policies can be validated and rerouted if needed, independent from the remote domain's head-end. Using binding segments isolates the head-end from topology changes in the remote domain.

Packets received with a BSID as top label are steered into the SR-TE policy associated with the BSID. When the BSID label is popped, the SR-TE policy's SID list is pushed.

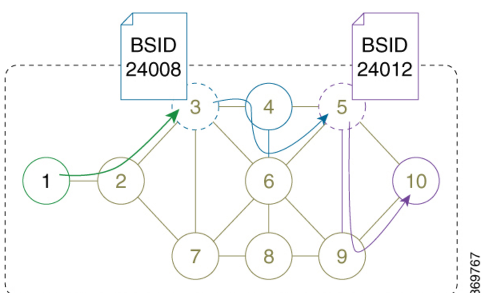
BSID can be used in the following cases:

- Multi-Domain (inter-domain, inter-autonomous system)—BSIDs can be used to steer traffic across domain borders, creating seamless end-to-end inter-domain SR-TE policies.
- Large-Scale within a single domain—The head-end can use hierarchical SR-TE policies by nesting the end-to-end (edge-to-edge) SR-TE policy within another layer of SR-TE policies (aggregation-to-aggregation). The SR-TE policies are nested within another layer of policies using the BSIDs, resulting in seamless end-to-end SR-TE policies.
- Label stack compression—If the label-stack size required for an SR-TE policy exceeds the platform capability, the SR-TE policy can be seamlessly stitched to, or nested within, other SR-TE policies using a binding segment.
- BGP SR-TE Dynamic—The head-end steers the packet into a BGP-based FIB entry whose next hop is a binding-SID.

# Stitching SR-TE Polices Using Binding SID: Example

In this intra-domain example, three SR-TE policies are stitched together to form a seamless end-to-end path from node 1 to node 10.

***Figure 3: Intra-Domain Topology***




**Step 1** Configure an SR-TE policy on node 5 to node 10 via node 9. Node 5 automatically allocates a binding-SID (24012) for the SR-TE policy.

**Example:**

```
RP/0/0/CPU0:xrvr-5(config)# explicit-path name PATH5-9_10
RP/0/0/CPU0:xrvr-5(config-expl-path)# index 10 next-address strict ipv4 unicast 192.168.59.9
RP/0/0/CPU0:xrvr-5(config-expl-path)# index 20 next-address strict ipv4 unicast 10.1.1.10
RP/0/0/CPU0:xrvr-5(config-expl-path)# exit

RP/0/0/CPU0:xrvr-5(config)# interface tunnel-te1
RP/0/0/CPU0:xrvr-5(config-if)# ipv4 unnumbered Loopback0
RP/0/0/CPU0:xrvr-5(config-if)# destination 10.1.1.10
RP/0/0/CPU0:xrvr-5(config-if)# path-option 1 explicit name PATH5-9_10 segment-routing
RP/0/0/CPU0:xrvr-5(config-if)# commit

RP/0/0/CPU0:xrvr-5# show mpls traffic-eng tunnels 1 detail
```

```
Name: tunnel-te1  Destination: 10.1.1.10  Ifhandle:0x680
  Signalled-Name: xrvr-5_t1
  Status:
    Admin:    up Oper:   up   Path:  valid   Signalling: connected
    path option 1, (Segment-Routing) type dynamic (Basis for Setup, path weight 10)
<...>
 Binding SID: 24012
<...>
    Segment-Routing Path Info (IS-IS 1 level-2)
      Segment0[Link]: 192.168.59.5 - 192.168.59.9, Label: 24007
      Segment1[Node]: 10.1.1.10, Label: 16010
```

**Step 2** Configure an SR-TE policy on node 3 to node 5 via node 4 and Link4-6, and push the binding-SID of the SR-TE policy at node 5 (24012) to stitch to the SR-TE policy on node 5. Node 3 automatically allocates a binding-SID (24008) for this SR-TE policy.

**Example:**

```
RP/0/0/CPU0:xrvr-3(config)# explicit-path name PATH4_4-6_5_BSID
RP/0/0/CPU0:xrvr-3(config-expl-path)# index 10 next-address strict ipv4 unicast 10.1.1.4
RP/0/0/CPU0:xrvr-3(config-expl-path)# index 20 next-address strict ipv4 unicast 192.168.46.6
RP/0/0/CPU0:xrvr-3(config-expl-path)# index 30 next-address strict ipv4 unicast 10.1.1.5
RP/0/0/CPU0:xrvr-3(config-expl-path)# index 40 next-label 24012
RP/0/0/CPU0:xrvr-3(config-expl-path)# exit

RP/0/0/CPU0:xrvr-3(config)# interface tunnel-te1
RP/0/0/CPU0:xrvr-3(config-if)# ipv4 unnumbered Loopback0
RP/0/0/CPU0:xrvr-3(config-if)# destination 10.1.1.10
RP/0/0/CPU0:xrvr-3(config-if)# path-option 1 explicit name PATH4_4-6_5_BSID segment-routing
RP/0/0/CPU0:xrvr-3(config-if)# commit

RP/0/0/CPU0:xrvr-3# show mpls traffic-eng tunnels 1 detail
Name: tunnel-te1  Destination: 10.1.1.10  Ifhandle:0x780
  Signalled-Name: xrvr-3_t1
  Status:
    Admin:    up Oper:   up   Path:  valid   Signalling: connected
    path option 1, (Segment-Routing) type explicit PATH4_6_5 (Basis for Setup)
<...>
 Binding SID: 24008
<...>
    Segment-Routing Path Info (IS-IS 1 level-2)
      Segment0[Node]: 10.1.1.4, Label: 16004
      Segment1[Link]: 192.168.46.4 - 192.168.46.6, Label: 24003
      Segment2[Node]: 10.1.1.5, Label: 16005
      Segment3[ - ]: Label: 24012
```

**Step 3** Configure an SR-TE policy on node 1 to node 3 and push the binding-SID of the SR-TE policy at node 3 (24008) to stitch to the SR-TE policy on node 3.

**Example:**

```
RP/0/0/CPU0:xrvr-1(config)# explicit-path name PATH3_BSID
RP/0/0/CPU0:xrvr-1(config-expl-path)# index 10 next-address strict ipv4 unicast 10.1.1.3
RP/0/0/CPU0:xrvr-1(config-expl-path)# index 20 next-label 24008
RP/0/0/CPU0:xrvr-1(config-expl-path)# exit

RP/0/0/CPU0:xrvr-1(config)# interface tunnel-te1
RP/0/0/CPU0:xrvr-1(config-if)# ipv4 unnumbered Loopback0
RP/0/0/CPU0:xrvr-1(config-if)# destination 10.1.1.10
RP/0/0/CPU0:xrvr-1(config-if)# path-option 1 explicit name PATH3_BSID segment-routing
```

```
RP/0/0/CPU0:xrvr-1(config-if)# commit

RP/0/0/CPU0:xrvr-1# show mpls traffic-eng tunnels 1 detail
Name: tunnel-te1  Destination: 10.1.1.10  Ifhandle:0x2f80
  Signalled-Name: xrvr-1_t1
  Status:
    Admin:    up Oper:   up   Path:  valid   Signalling: connected
    path option 1, (Segment-Routing) type explicit PATH3_BSID (Basis for Setup)
<...>
 Binding SID: 24002
<...>
    Segment-Routing Path Info (IS-IS 1 level-2)
      Segment0[Node]: 10.1.1.3, Label: 16003
      Segment1[ - ]: Label: 24008
```

The path is a chain of SR-TE policies stitched together using the binding-SIDs, providing a seamless end-to-end path.

```
RP/0/0/CPU0:xrvr-1# traceroute 10.1.1.10
Type escape sequence to abort.
Tracing the route to 10.1.1.10
 1  99.1.2.2 [MPLS: Labels 16003/24008 Exp 0] 29 msec  19 msec  19 msec
 2  99.2.3.3 [MPLS: Label 24008 Exp 0] 29 msec  19 msec  19 msec
 3  99.3.4.4 [MPLS: Labels 24003/16005/24012 Exp 0] 29 msec  19 msec  19 msec
 4  99.4.6.6 [MPLS: Labels 16005/24012 Exp 0] 29 msec  29 msec  19 msec
 5  99.5.6.5 [MPLS: Label 24012 Exp 0] 29 msec  29 msec  19 msec
 6  99.5.9.9 [MPLS: Label 16010 Exp 0] 19 msec  19 msec  19 msec
 7  99.9.10.10 29 msec  19 msec  19 msec
```

# Configure Seamless Bidirectional Forwarding Detection

Bidirectional forwarding detection (BFD) provides low-overhead, short-duration detection of failures in the path between adjacent forwarding engines. BFD allows a single mechanism to be used for failure detection over any media and at any protocol layer, with a wide range of detection times and overhead. The fast detection of failures provides immediate reaction to failure in the event of a failed link or neighbor.

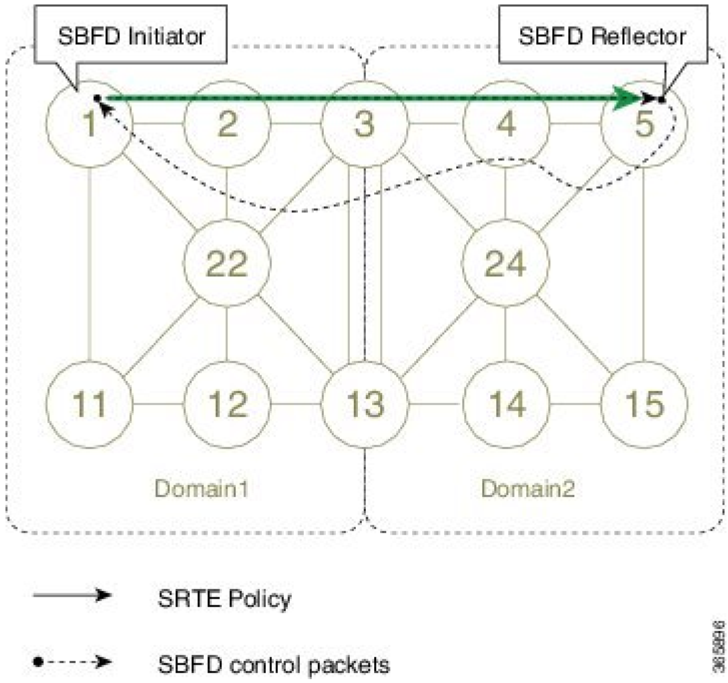
In BFD, each end of the connection maintains a BFD state and transmits packets periodically over a forwarding path. Seamless BFD (SBFD) is unidirectional, resulting in faster session activation than BFD. The BFD state and client context is maintained on the head-end (initiator) only. The tail-end (reflector) validates the BFD packet and responds, so there is no need to maintain the BFD state on the tail-end.

### Initiators and Reflectors

SBFD runs in an asymmetric behavior, using initiators and reflectors.

The following figure represents the roles of the SBFD initiator and reflector.

*Figure 4: SBFD Initiator and Reflector*




The initiator is an SBFD session on a network node that performs a continuity test to a remote entity by sending SBFD packets. The initiator injects the SBFD packets into the segment-routing traffic-engineering (SRTE) policy. The initiator triggers the SBFD session and maintains the BFD state and client context.

The reflector is an SBFD session on a network node that listens for incoming SBFD control packets to local entities and generates response SBFD control packets. The reflector is stateless and only reflects the SBFD packets back to the initiator.

A node can be both an initiator and a reflector, if you want to configure different SBFD sessions.

### Discriminators

The BFD control packet carries 32-bit discriminators (local and remote) to demultiplex BFD sessions. SBFD requires globally unique SBFD discriminators that are known by the initiator.

The SBFD control packets contain the discriminator of the initiator, which is created dynamically, and the discriminator of the reflector, which is configured as a local discriminator on the reflector.

# Configure the SBFD Reflector

To ensure the SBFD packet arrives on the intended reflector, each reflector has at least one globally unique discriminator. Globally unique discriminators of the reflector are known by the initiator before the session starts. An SBFD reflector only accepts BFD control packets where "Your Discriminator" is the reflector discriminator.

This task explains how to configure local discriminators on the reflector.

### Before you begin

Enable mpls oam on the reflector to install a routing information base (RIB) entry for 127.0.0.0/8.

```
Router_5# configure
Router_5(config)# mpls oam
Router_5(config-oam)#
```

### SUMMARY STEPS

1. **configure**
2. **sbfd**
3. **local-discriminator** {*ipv4-address* | *32-bit-value* | **dynamic** | **interface** *interface*}
4. **commit**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **sbfd**<br>**Example:**<br>`Router_5(config)# sbfd` | Enters SBFD configuration mode. |
| **Step 3** | **local-discriminator** {*ipv4-address* \| *32-bit-value* \| **dynamic** \| **interface** *interface*}<br>**Example:**<br>`Router_5(config-sbfd)# local-discriminator 1.1.1.5`<br>`Router_5(config-sbfd)# local-discriminator 987654321`<br>`Router_5(config-sbfd)# local-discriminator dynamic`<br>`Router_5(config-sbfd)# local-discriminator interface Loopback0` | Configures the local discriminator. You can configure multiple local discriminators. |
| **Step 4** | **commit** | |

Verify the local discriminator configuration.

#### Example

```
Router_5# show bfd target-identifier local

Local Target Identifier Table
-----------------------------
Discr        Discr Src   VRF       Status   Flags
                         Name
```

```
-----       ---------   -------   -------- --------
16843013    Local       default   enable   -----ia-
987654321   Local       default   enable   ----v---
2147483649  Local       default   enable   -------d

Legend: TID - Target Identifier
        a   - IP Address mode
        d   - Dynamic mode
        i   - Interface mode
        v   - Explicit Value mode
```

#### What to do next

Configure the SBFD initiator.

# Configure the SBFD Initiator

Perform the following configurations on the SBFD initiator.

## Enable Line Cards to Host BFD Sessions

The SBFD initiator sessions are hosted by the line card CPU.

This task explains how to enable line cards to host BFD sessions.

### SUMMARY STEPS

1. **configure**
2. **bfd**
3. **multipath include location** *node-id*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **bfd**<br>**Example:**<br>`Router_1(config)# bfd` | Enters BFD configuration mode. |
| **Step 3** | **multipath include location** *node-id*<br>**Example:**<br>`Router_1(config-bfd)# multipath include location 0/1/CPU0`<br>`Router_1(config-bfd)# multipath include location` | Configures BFD multiple path on specific line card. Any of the configured line cards can be instructed to host a BFD session. |

| | Command or Action | Purpose |
|---|---|---|
| | **0/2/CPU0**<br>`Router_1(config-bfd)#` **`multipath include location 0/3/CPU0`** | |

### What to do next

Map a destination address to a remote discriminator.

## Map a Destination Address to a Remote Discriminator

The SBFD initiator uses a Remote Target Identifier (RTI) table to map a destination address (Target ID) to a remote discriminator.

This task explains how to map a destination address to a remote discriminator.

### SUMMARY STEPS

1. **configure**
2. **sbfd**
3. **remote-target ipv4** *ipv4-address*
4. **remote-discriminator** *remote-discriminator*

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **sbfd**<br>**Example:**<br>`Router_1(config)#` **`sbfd`** | Enters SBFD configuration mode. |
| **Step 3** | **remote-target ipv4** *ipv4-address*<br>**Example:**<br>`Router_1(config-sbfd)#` **`remote-target ipv4 1.1.1.5`** | Configures the remote target. |
| **Step 4** | **remote-discriminator** *remote-discriminator*<br>**Example:**<br>`Router_1(config-sbfd-nnnn)#` **`remote-discriminator 16843013`** | Maps the destination address (Target ID) to a remote discriminator. |

Verify the remote discriminator configuration.

### Example

```
Router_1# show bfd target-identifier remote

Remote Target Identifier Table
------------------------------
Discr        Discr Src    VRF         TID Type    Status
             Target ID    Name
------       ---------    -------     --------    ------
16843013     Remote       default     ipv4        enable
             1.1.1.5

Legend: TID - Target Identifier
```

### What to do next

Enable SBFD on an SR-TE policy.

## Enable SBFD on the SR-TE Policy

This task explains how to enable SBFD on an SRTE policy.

### SUMMARY STEPS

1. **configure**
2. **interface tunnel-te** *tunnel-id*
3. **ipv4 unnumbered** *type interface-path-id*
4. **destination** *ip-address*
5. **bfd**
6. **fast-detect sbfd**
7. **multiplier** *multiplier*
8. **minimum-interval** *milliseconds*
9. **exit**
10. **path-option** *preference-priority* **explicit name** *path-name* **segment-routing**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **interface tunnel-te** *tunnel-id*<br>**Example:** | Configures the tunnel interface. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router_1(config)# interface tunnel-te1` | |
| **Step 3** | **ipv4 unnumbered** *type interface-path-id*<br>**Example:**<br>`Router_1(config-if)# ipv4 unnumbered loopback0` | Assigns a source address. |
| **Step 4** | **destination** *ip-address*<br>**Example:**<br>`Router_1(config-if)# destination 1.1.1.5` | Assigns a destination address. |
| **Step 5** | **bfd**<br>**Example:**<br>`Router_1(config-if)# bfd` | Enters BFD configuration mode. |
| **Step 6** | **fast-detect sbfd**<br>**Example:**<br>`Router_1(config-if-tunte-bfd)# fast-detect sbfd` | Enables SBFD fast-detection on the specified IPV4 destination address. |
| **Step 7** | **multiplier** *multiplier*<br>**Example:**<br>`Router_1(config-if-tunte-bfd)# multiplier 2` | Sets the number of times a packet is missed before BFD declares the neighbor down. The range is from 2 to 10. The default is 3. |
| **Step 8** | **minimum-interval** *milliseconds*<br>**Example:**<br>`Router_1(config-if-tunte-bfd)# minimum-interval 50` | Sets the interval between sending BFD hello packets to the neighbor. The range is from 15 to 200. The default is 15. |
| **Step 9** | **exit**<br>**Example:**<br>`Router_1(config-if-tunte-bfd)# exit` | |
| **Step 10** | **path-option** *preference-priority* **explicit name** *path-name* **segment-routing**<br>**Example:** | Specifies the explicit path name and assigns the path ID. |

| | Command or Action | Purpose |
|---|---|---|
| | `Router_1(config-if)# path-option 1 explicit name r1r6_exp segment-routing` | |

The SBFD initiator looks in the RTI table for a remote-discriminator value mapped to destination address (1.1.1.5). If there is no RTI entry for 1.1.1.5, then the SBFD initiator uses 1.1.1.5 as the remote-discriminator value.

CHAPTER 8

# Configure Segment Routing Path Computation Element

The Segment Routing Path Computation Element (SR-PCE) provides stateful PCE functionality by extending the existing IOS-XR PCEP functionality with additional capabilities. SR-PCE is supported on the MPLS data plane and IPv4 control plane.

> **Note** The Cisco IOS XRv 9000 is the recommended platform to act as the SR-PCE. Refer to the Cisco IOS XRv 9000 Router Installation and Configuration Guide for more information.



# About SR-PCE

The path computation element protocol (PCEP) describes a set of procedures by which a path computation client (PCC) can report and delegate control of head-end label switched paths (LSPs) sourced from the PCC to a PCE peer. The PCE can request the PCC to update and modify parameters of LSPs it controls. The stateful model also enables a PCC to allow the PCE to initiate computations allowing the PCE to perform network-wide orchestration.

> **Note** For more information on PCE, PCC, and PCEP, refer to the Path Computation Element section in the *MPLS Configuration Guide for Cisco ASR 9000 Series Routers*.

SR-PCE learns topology information by way of IGP (OSPF or IS-IS) or through BGP Link-State (BGP-LS).

SR-PCE is capable of computing paths using the following methods:

- TE metric—SR-PCE uses the TE metric in its path calculations to optimize cumulative TE metric.
- IGP metric—SR-PCE uses the IGP metric in its path calculations to optimize reachability.
- LSP Disjointness—SR-PCE uses the path computation algorithms to compute a pair of disjoint LSPs. The disjoint paths can originate from the same head-end or different head-ends. Disjoint level refers to

the type of resources that should not be shared by the two computed paths. SR-PCE supports the following disjoint path computations:

- Link – Specifies that links are not shared on the computed paths.
- Node – Specifies that nodes are not shared on the computed paths.
- SRLG – Specifies that links with the same SRLG value are not shared on the computed paths.
- SRLG-node – Specifies that SRLG and nodes are not shared on the computed paths.

When the first request is received with a given disjoint-group ID, the first LSP is computed, encoding the shortest path from the first source to the first destination. When the second LSP request is received with the same disjoint-group ID, information received in both requests is used to compute two disjoint paths: one path from the first source to the first destination, and another path from the second source to the second destination. Both paths are computed at the same time.

# Configure SR-PCE

This task explains how to configure SR-PCE.

### Before you begin

The Cisco IOS XRv 9000 is the recommended platform to act as the SR-PCE.

### SUMMARY STEPS

1. **configure**
2. **pce**
3. **address ipv4** *address*
4. **state-sync ipv4** *address*
5. **tcp-buffer size** *size*
6. **password** {**clear** | **encrypted**} *password*
7. **segment-routing** {**strict-sid-only** | **te-latency**}
8. **timers**
9. **keepalive** *time*
10. **minimum-peer-keepalive** *time*
11. **reoptimization** *time*
12. **exit**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **pce**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# pce` | Enables PCE and enters PCE configuration mode. |
| **Step 3** | **address ipv4** *address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pce)# address ipv4 192.168.0.1` | Configures a PCE IPv4 address. |
| **Step 4** | **state-sync ipv4** *address*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pce)# state-sync ipv4 192.168.0.3` | Configures the remote peer for state synchronization. |
| **Step 5** | **tcp-buffer size** *size*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pce)# tcp-buffer size 1024000` | Configures the transmit and receive TCP buffer size for each PCEP session, in bytes. The default buffer size is 256000. The valid range is from 204800 to 1024000. |
| **Step 6** | **password** {**clear** \| **encrypted**} *password*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pce)# password encrypted pwd1` | Enables TCP MD5 authentication for all PCEP peers. Any TCP segment coming from the PCC that does not contain a MAC matching the configured password will be rejected. Specify if the password is encrypted or clear text. |
| **Step 7** | **segment-routing** {**strict-sid-only** \| **te-latency**}<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pce)# segment-routing strict-sid-only` | Configures the segment routing algorithm to use strict SID or TE latency.<br>**Note** This setting is global and applies to all LSPs that request a path from this controller. |
| **Step 8** | **timers**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-pce)# timers` | Enters timer configuration mode. |
| **Step 9** | **keepalive** *time*<br>**Example:** | Configures the timer value for locally generated keep-alive messages. The default time is 30 seconds. |

| | Command or Action | Purpose |
|---|---|---|
| | RP/0/RSP0/CPU0:router(config-pce-timers)# **keepalive 60** | |
| **Step 10** | **minimum-peer-keepalive** *time*<br>**Example:**<br>RP/0/RSP0/CPU0:router(config-pce-timers)# **minimum-peer-keepalive 30** | Configures the minimum acceptable keep-alive timer that the remote peer may propose in the PCEP OPEN message during session establishment. The default time is 20 seconds. |
| **Step 11** | **reoptimization** *time*<br>**Example:**<br>RP/0/RSP0/CPU0:router(config-pce-timers)# **reoptimization 600** | Configures the re-optimization timer. The default timer is 1800 seconds. |
| **Step 12** | **exit**<br>**Example:**<br>RP/0/RSP0/CPU0:router(config-pce-timers)# **exit** | Exits timer configuration mode and returns to PCE configuration mode. |

# Configure the Disjoint Policy (Optional)

This task explains how to configure the SR-PCE to compute disjointness for a pair of LSPs signaled by PCCs that do not include the PCEP association group-ID object in their PCEP request. This can be beneficial for deployments where PCCs do not support this PCEP object or when the network operator prefers to manage the LSP disjoint configuration centrally.

### SUMMARY STEPS

1. **disjoint-path**
2. **group-id** *value* **type** {**link** | **node** | **srlg** | **srlg-node**} [**sub-id** *value*]
3. **strict**
4. **lsp** {**1** | **2**} **pcc ipv4** *address* **lsp-name** *lsp_name* [**shortest-path**]

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **disjoint-path**<br>**Example:**<br>RP/0/RSP0/CPU0:router(config-pce)# **disjoint-path** | Enters disjoint configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **group-id** *value* **type** {**link** \| **node** \| **srlg** \| **srlg-node**} [**sub-id** *value*]<br><br>**Example:**<br><br>`RP/0/RSP0/CPU0:router(config-pce-disjoint)#` **`group-id 1 type node sub-id 1`** | Configures the disjoint group ID and defines the preferred level of disjointness (the type of resources that should not be shared by the two paths):<br><br>• **link**—Specifies that links are not shared on the computed paths.<br>• **node**—Specifies that nodes are not shared on the computed paths.<br>• **srlg**—Specifies that links with the same SRLG value are not shared on the computed paths.<br>• **srlg-node**—Specifies that SRLG and nodes are not shared on the computed paths.<br><br>If a pair of paths that meet the requested disjointness level cannot be found, then the paths will automatically fallback to a lower level:<br><br>• If the requested disjointness level is SRLG or node, then link-disjoint paths will be computed.<br>• If the requested disjointness level was link, or if the first fallback from SRLG or node disjointness failed, then the lists of segments encoding two shortest paths, without any disjointness constraint, will be computed. |
| **Step 3** | **strict**<br><br>**Example:**<br><br>`RP/0/RSP0/CPU0:router(config-pce-disjoint)#` **`strict`** | (Optional) Prevents the automatic fallback behavior of the preferred level of disjointness. If a pair of paths that meet the requested disjointness level cannot be found, the disjoint calculation terminates and no new path is provided. The existing path is not modified. |
| **Step 4** | **lsp** {**1** \| **2**} **pcc ipv4** *address* **lsp-name** *lsp_name* [**shortest-path**]<br><br>**Example:**<br><br>`RP/0/RSP0/CPU0:router(config-pce-disjoint)#` **`lsp 1 pcc ipv4 192.168.0.1 lsp-name rtrA_t1 shortest-path`**<br>`RP/0/RSP0/CPU0:router(config-pce-disjoint)#` **`lsp 2 pcc ipv4 192.168.0.5 lsp-name rtrE_t2`** | Adds LSPs to the disjoint group.<br><br>The **shortest-path** keyword forces one of the disjoint paths to follow the shortest path from the source to the destination. This option can only be applied to the the first LSP specified. |

CHAPTER 9

# Configure Topology-Independent Loop-Free Alternate (TI-LFA)

Topology-Independent Loop-Free Alternate (TI-LFA) uses segment routing to provide link, node, and Shared Risk Link Groups (SRLG) protection in topologies where other fast reroute techniques cannot provide protection.

- Classic Loop-Free Alternate (LFA) is topology dependent, and therefore cannot protect all destinations in all networks. A limitation of LFA is that, even if one or more LFAs exist, the optimal LFA may not always be provided.
- Remote LFA (RLFA) extends the coverage to 90-95% of the destinations, but it also does not always provide the most desired repair path. RLFA also adds more operational complexity by requiring a targeted LDP session to the RLFAs to protect LDP traffic.

TI-LFA provides a solution to these limitations while maintaining the simplicity of the IPFRR solution.

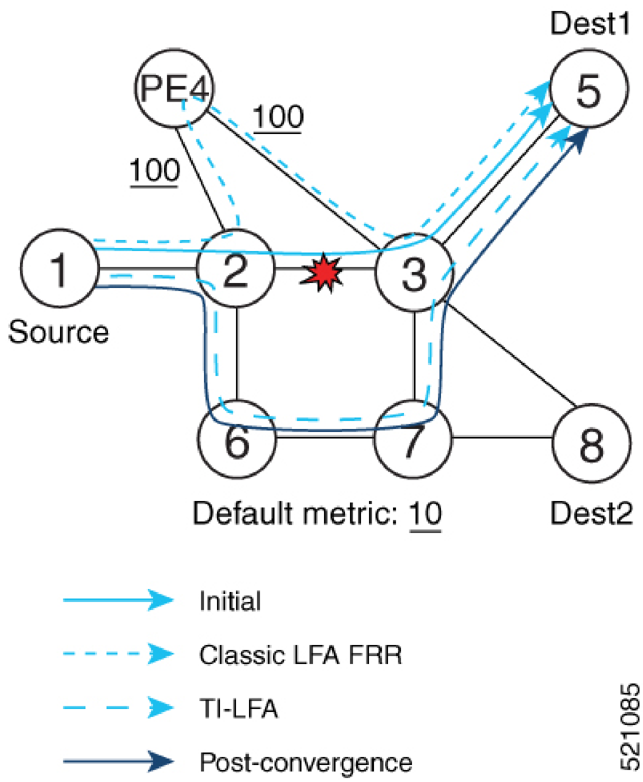
The goal of TI-LFA is to reduce the packet loss that results while routers converge after a topology change due to a link or node failure. Rapid failure repair (< 50 msec) is achieved through the use of pre-calculated backup paths that are loop-free and safe to use until the distributed network convergence process is completed.

The optimal repair path is the path that the traffic will eventually follow after the IGP has converged. This is called the post-convergence path. This path is preferred for the following reasons:

- Optimal for capacity planning — During the capacity-planning phase of the network, the capacity of a link is provisioned while taking into consideration that such link with be used when other links fail.
- Simple to operate — There is no need to perform a case-by-case adjustments to select the best LFA among multiple candidate LFAs.
- Fewer traffic transitions — Since the repair path is equal to the post-convergence path, the traffic switches paths only once.

The following topology illustrates the optimal and automatic selection of the TI-LFA repair path.

*Figure 5: TI-LFA Repair Path*




Node 2 protects traffic to destination Node 5.

With classic LFA, traffic would be steered to Node 4 after a failure of the protected link. This path is not optimal, since traffic is routed over edge node Node 4 that is connected to lower capacity links.

TI-LFA calculates a post-convergence path and derives the segment list required to steer packets along the post-convergence path without looping back.

In this example, if the protected link fails, the shortest path from Node2 to Node5 would be:

Node2 → Node6 → Node7 → Node3 → Node5

Node7 is the PQ-node for destination Node5. TI-LFA encodes a single segment (prefix SID of Node7) in the header of the packets on the repair path.

### TI-LFA Protection Types

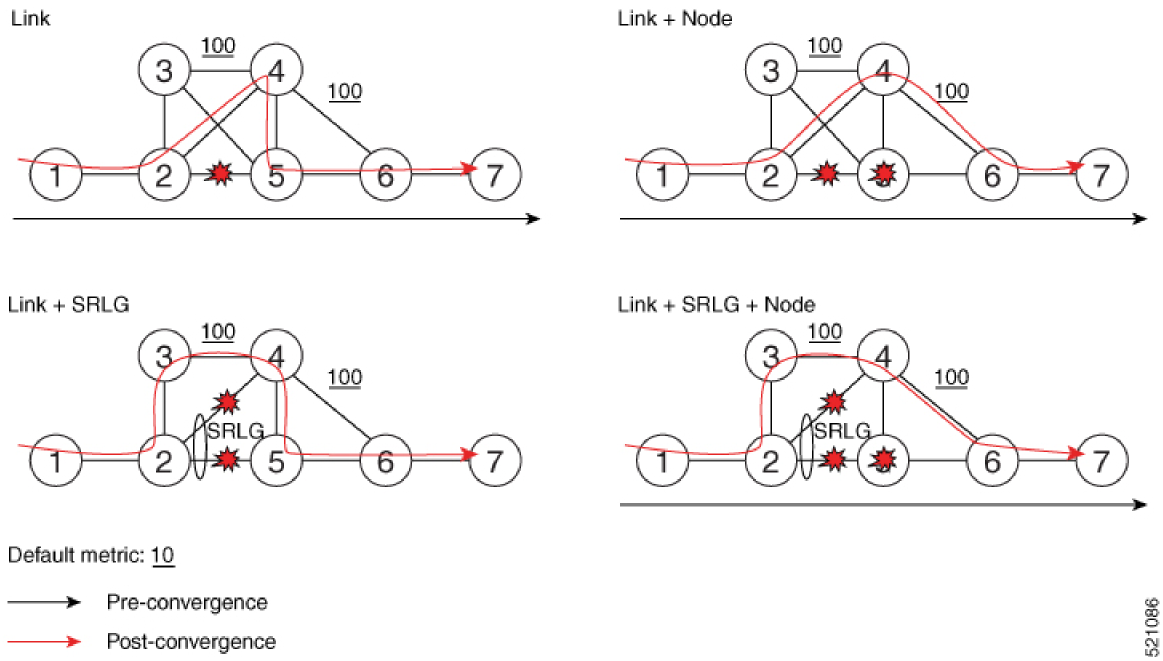
TI-LFA supports the following protection:

- Link protection — The link is excluded during the post-convergence backup path calculation.
- Node protection — The neighbor node is excluded during the post convergence backup path calculation.
- Shared Risk Link Groups (SRLG) protection — SRLG refer to situations in which links in a network share a common fiber (or a common physical attribute). These links have a shared risk: when one link fails, other links in the group might also fail. TI-LFA SRLG protection attempts to find the post-convergence backup path that excludes the SRLG of the protected link. All local links that share any SRLG with the protecting link are excluded.

When you enable link protection, you can also enable node protection, SRLG protection, or both, and specify a tiebreaker priority in case there are multiple LFAs.

The following example illustrates the link, node, and SRLG protection types. In this topology, Node2 applies different protection models to protect traffic to Node7.

*Figure 6: TI-LFA Protection Types*






# Behaviors and Limitations of TI-LFA

The behaviors and limitations of TI-LFA are listed below:

- TI-LFA protects unlabeled IPv4 traffic.
- TI-LFA does not protect unlabeled IPv6 traffic.
- The Network Virtualization (nV) satellite access interfaces do not support TI-LFA.
- Bridge Virtual Interfaces (BVI) do not support TI-LFA.

# Configuring TI-LFA for IS-IS

This task describes how to enable per-prefix Topology Independent Loop-Free Alternate (TI-LFA) computation to converge traffic flows around link, node, and SRLG failures.

### Before you begin

Ensure that the following topology requirements are met:

- Router interfaces are configured as per the topology.
- Routers are configured with IS-IS.
- Segment routing for IS-IS is configured. See Enabling Segment Routing for IS-IS Protocol, on page 11.
- Enter the following commands in global configuration mode:

```
Router(config)# ipv4 unnumbered mpls traffic-eng Loopback0
Router(config)# mpls traffic-eng
Router(config-mpls-te)# exit
Router(config)#
```

## SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **interface** *type interface-path-id*
4. **address-family ipv4** [**unicast**]
5. **fast-reroute per-prefix**
6. **fast-reroute per-prefix ti-lfa**
7. **fast-reroute per-prefix tiebreaker** {**node-protecting** | **srlg-disjoint**} **index** *priority*

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router isis** *instance-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router isis 1` | Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.<br>**Note** You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command. |
| **Step 3** | **interface** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet0/0/2/1`<br>`RP/0/RSP0/CPU0:router(config-isis)# interface Bundle-Ether1` | Enters interface configuration mode.<br>**Note** You can configure TI-LFA under Ethernet-based interfaces and logical Bundle-Ethernet interfaces. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **address-family ipv4** [**unicast**]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast` | Specifies the IPv4 address family, and enters router address family configuration mode. |
| **Step 5** | **fast-reroute per-prefix**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix` | Enables per-prefix fast reroute. |
| **Step 6** | **fast-reroute per-prefix ti-lfa**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa` | Enables per-prefix TI-LFA fast reroute link protection. |
| **Step 7** | **fast-reroute per-prefix tiebreaker** {**node-protecting** \| **srlg-disjoint**} **index** *priority*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix tie-breaker srlg-disjoint index 100` | Enables TI-LFA node or SRLG protection and specifies the tiebreaker priority. Valid *priority* values are from 1 to 255. The lower the *priority* value, the higher the priority of the rule. Link protection always has a lower priority than node or SRLG protection.<br>**Note** The same attribute cannot be configured more than once on an interface.<br>**Note** For IS-IS, TI-LFA node protection and SRLG protection can be configured on the interface or the instance. |

TI-LFA has been successfully configured for segment routing.

# Configuring TI-LFA for OSPF

This task describes how to enable per-prefix Topology Independent Loop-Free Alternate (TI-LFA) computation to converge traffic flows around link, node, and SRLG failures.

**Note** TI-LFA can be configured on the instance, area, or interface. When configured on the instance or area, all interfaces in the instance or area inherit the configuration.

### Before you begin

Ensure that the following topology requirements are met:

- Router interfaces are configured as per the topology.
- Routers are configured with OSPF.
- Segment routing for OSPF is configured. See Enabling Segment Routing for OSPF Protocol, on page 25.
- Enter the following commands in global configuration mode:

```
Router(config)# ipv4 unnumbered mpls traffic-eng Loopback0
Router(config)# mpls traffic-eng
Router(config-mpls-te)# exit
Router(config)#
```

## SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **area** *area-id*
4. **interface** *type interface-path-id*
5. **fast-reroute per-prefix**
6. **fast-reroute per-prefix ti-lfa**
7. **fast-reroute per-prefix tiebreaker** {**node-protecting** | **srlg-disjoint**} **index** *priority*

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router ospf** *process-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router ospf 1` | Enables OSPF routing for the specified routing process, and places the router in router configuration mode. |
| **Step 3** | **area** *area-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)# area 1` | Enters area configuration mode. |
| **Step 4** | **interface** *type interface-path-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/0/2/1`<br>`RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Bundle-Ether1` | Enters interface configuration mode.<br>**Note** You can configure TI-LFA under Ethernet-based interfaces and logical Bundle-Ethernet interfaces. |

| | Command or Action | Purpose |
|---|---|---|
| Step 5 | **fast-reroute per-prefix**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar-if)#` **`fast-reroute per-prefix`** | Enables per-prefix fast reroute. |
| Step 6 | **fast-reroute per-prefix ti-lfa**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar-if)#` **`fast-reroute per-prefix ti-lfa`** | Enables per-prefix TI-LFA fast reroute link protection. |
| Step 7 | **fast-reroute per-prefix tiebreaker** {**node-protecting** \| **srlg-disjoint**} **index** *priority*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf-ar-if)#` **`fast-reroute per-prefix tie-breaker srlg-disjoint index 100`** | Enables TI-LFA node or SRLG protection and specifies the tiebreaker priority. Valid *priority* values are from 1 to 255. The lower the *priority* value, the higher the priority of the rule. Link protection always has a lower priority than node or SRLG protection.<br>**Note** The same attribute cannot be configured more than once on an interface. |

TI-LFA has been successfully configured for segment routing.

# TI-LFA Node and SRLG Protection: Examples

The following examples show the configuration of the tiebreaker priority for TI-LFA node and SRLG protection, and the behavior of post-convergence backup-path. These examples use OSPF, but the same configuration and behavior applies to IS-IS.

### Example: Enable link-protecting and node-protecting TI-LFA

```
router ospf 1
 area 1
  interface GigabitEthernet0/0/2/1
    fast-reroute per-prefix
    fast-reroute per-prefix ti-lfa
    fast-reroute per-prefix tiebreaker node-protecting index 100
```

Both link-protecting and node-protecting TI-LFA backup paths will be computed. If the priority associated with the node-protecting tiebreaker is higher than any other tiebreakers, then node-protecting post-convergence backup paths will be selected, if it is available.

### Example: Enable link-protecting and SRLG-protecting TI-LFA

```
router ospf 1
 area 1
  interface GigabitEthernet0/0/2/1
    fast-reroute per-prefix
    fast-reroute per-prefix ti-lfa
```

```
    fast-reroute per-prefix tiebreaker srlg-disjoint index 100
```

Both link-protecting and SRLG-protecting TI-LFA backup paths will be computed. If the priority associated with the SRLG-protecting tiebreaker is higher than any other tiebreakers, then SRLG-protecting post-convergence backup paths will be selected, if it is available.

### Example: Enable link-protecting, node-protecting and SRLG-protecting TI-LFA

```
router ospf 1
 area 1
  interface GigabitEthernet0/0/2/1
    fast-reroute per-prefix
    fast-reroute per-prefix ti-lfa
    fast-reroute per-prefix tiebreaker node-protecting index 100
    fast-reroute per-prefix tiebreaker srlg-disjoint index 200
```

Link-protecting, node-protecting, and SRLG-protecting TI-LFA backup paths will be computed. If the priority associated with the node-protecting tiebreaker is highest from all tiebreakers, then node-protecting post-convergence backup paths will be selected, if it is available. If the node-protecting backup path is not available, SRLG-protecting post-convergence backup path will be used, if it is available.

CHAPTER 10

# Configure Segment Routing Microloop Avoidance

The Segment Routing Microloop Avoidance feature enables link-state routing protocols, such as IS-IS, to prevent or avoid microloops during network convergence after a topology change.



## About Segment Routing Microloop Avoidance

Microloops are brief packet loops that occur in the network following a topology change (link down, link up, or metric change events). Microloops are caused by the non-simultaneous convergence of different nodes in the network. If nodes converge and send traffic to a neighbor node that has not converged yet, traffic may be looped between these two nodes, resulting in packet loss, jitter, and out-of-order packets.

The Segment Routing Microloop Avoidance feature detects if microloops are possible following a topology change. If a node computes that a microloop could occur on the new topology, the node creates a loop-free SR-TE policy path to the destination using a list of segments. After the RIB update delay timer expires, the SR-TE policy is replaced with regular forwarding paths.

## Segment Routing Microloop Avoidance Limitations

For IS-IS, Segment Routing Microloop Avoidance is not supported when incremental shortest path first (ISPF) is configured.

## Configure Segment Routing Microloop Avoidance for IS-IS

This task describes how to enable Segment Routing Microloop Avoidance and set the Routing Information Base (RIB) update delay value for IS-IS.

### Before you begin

Ensure that the following topology requirements are met:

- Router interfaces are configured as per the topology.

- Routers are configured with IS-IS.
- Segment routing for IS-IS is configured. See Enabling Segment Routing for IS-IS Protocol, on page 11.
- Enter the following commands in global configuration mode:

```
Router(config)# ipv4 unnumbered mpls traffic-eng Loopback0
Router(config)# mpls traffic-eng
Router(config-mpls-te)# exit
Router(config)#
```

## SUMMARY STEPS

1. **configure**
2. **router isis** *instance-id*
3. **address-family ipv4** [ **unicast** ]
4. **microloop avoidance segment-routing**
5. **microloop avoidance rib-update-delay** *delay-time*

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router isis** *instance-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router isis 1` | Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.<br>You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command. |
| **Step 3** | **address-family ipv4** [ **unicast** ]<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast` | Specifies the IPv4 address family and enters router address family configuration mode. |
| **Step 4** | **microloop avoidance segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# microloop avoidance segment-routing` | Enables Segment Routing Microloop Avoidance. |
| **Step 5** | **microloop avoidance rib-update-delay** *delay-time*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# microloop avoidance rib-update-delay 3000` | Specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The *delay-time* is in milliseconds. The range is from 1-60000. The default value is 5000. |

# Configure Segment Routing Microloop Avoidance for OSPF

This task describes how to enable Segment Routing Microloop Avoidance and set the Routing Information Base (RIB) update delay value for OSPF.

### Before you begin

Ensure that the following topology requirements are met:

- Router interfaces are configured as per the topology.
- Routers are configured with OSPF.
- Segment routing for OSPF is configured. See Enabling Segment Routing for OSPF Protocol, on page 25.
- Enter the following commands in global configuration mode:

```
Router(config)# ipv4 unnumbered mpls traffic-eng Loopback0
Router(config)# mpls traffic-eng
Router(config-mpls-te)# exit
Router(config)#
```

## SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **microloop avoidance segment-routing**
4. **microloop avoidance rib-update-delay** *delay-time*

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **router ospf** *process-name*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router ospf 1` | Enables OSPF routing for the specified routing process, and places the router in router configuration mode. |
| **Step 3** | **microloop avoidance segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)# microloop avoidance segment-routing` | Enables Segment Routing Microloop Avoidance. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **microloop avoidance rib-update-delay** *delay-time*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-ospf)#` **`microloop avoidance rib-update-delay 3000`** | Specifies the amount of time the node uses the microloop avoidance policy before updating its forwarding table. The *delay-time* is in milliseconds. The range is from 1-60000. The default value is 5000. |

CHAPTER 11

# Configure Segment Routing Mapping Server

The mapping server is a key component of the interworking between LDP and segment routing. It enables SR-capable nodes to interwork with LDP nodes. The mapping server advertises Prefix-to-SID mappings in IGP on behalf of other non-SR-capable nodes.



## Segment Routing Mapping Server

The mapping server functionality in Cisco IOS XR segment routing centrally assigns prefix-SIDs for some or all of the known prefixes. A router must be able to act as a mapping server, a mapping client, or both.

- A router that acts as a mapping server allows the user to configure SID mapping entries to specify the prefix-SIDs for some or all prefixes. This creates the local SID-mapping policy. The local SID-mapping policy contains non-overlapping SID-mapping entries. The mapping server advertises the local SID-mapping policy to the mapping clients.
- A router that acts as a mapping client receives and parses remotely received SIDs from the mapping server to create remote SID-mapping entries.
- A router that acts as a mapping server and mapping client uses the remotely learnt and locally configured mapping entries to construct the non-overlapping consistent active mapping policy. IGP instance uses the active mapping policy to calculate the prefix-SIDs of some or all prefixes.

The mapping server automatically manages the insertions and deletions of mapping entries to always yield an active mapping policy that contains non-overlapping consistent SID-mapping entries.

- Locally configured mapping entries must not overlap each other.
- The mapping server takes the locally configured mapping policy, as well as remotely learned mapping entries from a particular IGP instance, as input, and selects a single mapping entry among overlapping mapping entries according to the preference rules for that IGP instance. The result is an active mapping policy that consists of non-overlapping consistent mapping entries.
- At steady state, all routers, at least in the same area or level, must have identical active mapping policies.

## Segment Routing Mapping Server Restrictions

- The position of the mapping server in the network is not important. However, since the mapping advertisements are distributed in IGP using the regular IGP advertisement mechanism, the mapping server needs an IGP adjacency to the network.
- The role of the mapping server is crucial. For redundancy purposes, you should configure multiple mapping servers in the networks.
- The mapping server functionality does not support a scenario where SID-mapping entries learned through one IS-IS instance are used by another IS-IS instance to determine the prefix-SID of a prefix. For example, mapping entries learnt from remote routers by 'router isis 1' cannot be used to calculate prefix-SIDs for prefixes learnt, advertised, or downloaded to FIB by 'router isis 2'. A mapping server is required for each IS-IS area.
- Segment Routing Mapping Server does not support Virtual Routing and Forwarding (VRF) currently.

# Segment Routing and LDP Interoperability

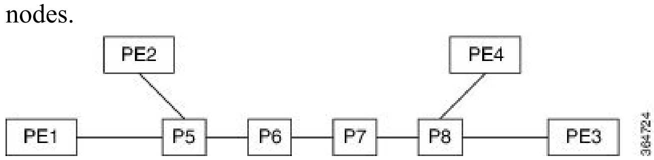
IGP provides mechanisms through which segment routing (SR) interoperate with label distribution protocol (LDP). The control plane of segment routing co-exists with LDP.

The Segment Routing Mapping Server (SRMS) functionality in SR is used to advertise SIDs for destinations, in the LDP part of the network, that do not support SR. SRMS maintains and advertises segment identifier (SID) mapping entries for such destinations. IGP propagates the SRMS mapping entries and interacts with SRMS to determine the SID value when programming the forwarding plane. IGP installs prefixes and corresponding labels, into routing information base (RIB), that are used to program the forwarding information base (FIB).

## Example: Segment Routing LDP Interoperability

Consider a network with a mix of segment routing (SR) and label distribution protocol (LDP). A continuous multiprotocol label switching (MPLS) LSP (Labeled Switched Path) can be established by facilitating interoperability. One or more nodes in the SR domain act as segment routing mapping server (SRMS). SRMS advertises SID mappings on behalf of non-SR capable nodes. Each SR-capable node learns about SID assigned to non-SR capable nodes without explicitly configuring individual nodes.

Consider a network as shown in the following image. This network is a mix of both LDP and SR-capable nodes.




In this mixed network:

- Nodes P6, P7, P8, PE4 and PE3 are LDP-capable
- Nodes PE1, PE2, P5 and P6 are SR-capable
- Nodes PE1, PE2, P5 and P6 are configured with segment routing global block (SRGB) of (100, 200)
- Nodes PE1, PE2, P5 and P6 are configured with node segments of 101, 102, 105 and 106 respectively

A service flow must be established from PE1 to PE3 over a continuous MPLS tunnel. This requires SR and LDP to interoperate.

### LDP to SR

The traffic flow from LDP to SR (right to left) involves:

1. PE3 learns a service route whose nhop is PE1. PE3 has an LDP label binding from the nhop P8 for the FEC PE1. PE3 forwards the packet P8.
2. P8 has an LDP label binding from its nhop P7 for the FEC PE1. P8 forwards the packet to P7.
3. P7 has an LDP label binding from its nhop P6 for the FEC PE1. P7 forwards the packet to P6.
4. P6 does not have an LDP binding from its nhop P5 for the FEC PE1. But P6 has an SR node segment to the IGP route PE1. P6 forwards the packet to P5 and swaps its local LDP label for FEC PE1 by the equivalent node segment 101. This process is called label merging.
5. P5 pops 101, assuming PE1 has advertised its node segment 101 with the penultimate-pop flag set and forwards to PE1.
6. PE1 receives the tunneled packet and processes the service label.

The end-to-end MPLS tunnel is established from an LDP LSP from PE3 to P6 and the related node segment from P6 to PE1.

### SR to LDP

Suppose that the operator configures P5 as a Segment Routing Mapping Server (SRMS) and advertises the mappings (P7, 107), (P8, 108), (PE3, 103) and (PE4, 104). If PE3 was SR-capable, the operator may have configured PE3 with node segment 103. Because PE3 is non-SR capable, the operator configures that policy at the SRMS; the SRMS advertises the mapping on behalf of the non-SR capable nodes. Multiple SRMS servers can be provisioned in a network for redundancy. The mapping server advertisements are only understood by the SR-capable nodes. The SR capable routers install the related node segments in the MPLS data plane in exactly the same manner if node segments were advertised by the nodes themselves.

The traffic flow from SR to LDP (left to right) involves:

1. PE1 installs the node segment 103 with nhop P5 in exactly the same manner if PE3 had advertised node segment 103.
2. P5 swaps 103 for 103 and forwards to P6.
3. The nhop for P6 for the IGP route PE3 is non-SR capable. (P7 does not advertise the SR capability.) However, P6 has an LDP label binding from that nhop for the same FEC. (For example, LDP label 103.) P6 swaps 103 for 103 and forwards to P7. We refer to this process as label merging.
4. P7 swaps this label with the LDP label received from P8 and forwards to P8.
5. P8 pops the LDP label and forwards to PE3.
6. PE3 receives the packet and processes as required.

The end-to-end MPLS LSP is established from an SR node segment from PE1 to P6 and an LDP LSP from P6 to PE3.

# Configuring Mapping Server

Perform these tasks to configure the mapping server and to add prefix-SID mapping entries in the active local mapping policy.

### SUMMARY STEPS

1. **configure**
2. **segment-routing**
3. **mapping-server**
4. **prefix-sid-map**
5. **address-family ipv4 | ipv6**
6. *ip-address / prefix-length first-SID-value* **range** *range*
7. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **segment-routing**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# segment-routing` | Enables segment routing. |
| **Step 3** | **mapping-server**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-sr)# mapping-server` | Enables mapping server configuration mode. |
| **Step 4** | **prefix-sid-map**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-sr-ms)# prefix-sid-map` | Enables prefix-SID mapping configuration mode.<br>**Note** Two-way prefix SID can be enabled directly under IS-IS or through a mapping server. |
| **Step 5** | **address-family ipv4 \| ipv6**<br>**Example:**<br>This example shows the address-family for ipv4:<br>`RP/0/RSP0/CPU0:router(config-sr-ms-map)# address-family ipv4` | Configures address-family for IS-IS. |

| | Command or Action | Purpose |
|---|---|---|
| | This example shows the address-family for ipv6:<br><br>`RP/0/RSP0/CPU0:router(config-sr-ms-map)#` **`address-family ipv6`** | |
| **Step 6** | *ip-address / prefix-length first-SID-value* **range** *range*<br>**Example:**<br><br>`RP/0/RSP0/CPU0:router(config-sr-ms-map-af)#` **`10.1.1.1/32 10 range 200`**<br>`RP/0/RSP0/CPU0:router(config-sr-ms-map-af)#` **`20.1.0.0/16 400 range 300`** | Adds SID-mapping entries in the active local mapping policy. In the configured example:<br>• Prefix 10.1.1.1/32 is assigned prefix-SID 10, prefix 10.1.1.2/32 is assigned prefix-SID 11,…, prefix 10.1.1.199/32 is assigned prefix-SID 200<br>• Prefix 20.1.0.0/16 is assigned prefix-SID 400, prefix 20.2.0.0/16 is assigned prefix-SID 401,…, and so on. |
| **Step 7** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify information about the locally configured prefix-to-SID mappings.

**Note** Specify the address family for IS-IS.

```
RP/0/RSP0/CPU0:router# show segment-routing mapping-server prefix-sid-map ipv4
Prefix               SID Index    Range        Flags
20.1.1.0/24          400          300
10.1.1.1/32          10           200

Number of mapping entries: 2

RP/0/RSP0/CPU0:router# show segment-routing mapping-server prefix-sid-map ipv4 detail
Prefix
20.1.1.0/24
    SID Index:      400
    Range:          300
    Last Prefix:    20.2.44.0/24
    Last SID Index: 699
    Flags:
10.1.1.1/32
    SID Index:      10
    Range:          200
    Last Prefix:    10.1.1.200/32
    Last SID Index: 209
```

```
    Flags:

Number of mapping entries: 2
```

### What to do next

Enable the advertisement of the local SID-mapping policy in the IGP.

# Enable Mapping Advertisement

In addition to configuring the static mapping policy, you must enable the advertisement of the mappings in the IGP.

Perform these steps to enable the IGP to advertise the locally configured prefix-SID mapping.

## Configure Mapping Advertisement for IS-IS

### SUMMARY STEPS

1. **router isis** *instance-id*
2. **address-family** { **ipv4** | **ipv6** } [ **unicast** ]
3. **segment-routing prefix-sid-map advertise-local**
4. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **router isis** *instance-id*<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config)# router isis 1` | Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.<br>• You can change the level of routing to be performed by a particular routing instance by using the **is-type** router configuration command. |
| **Step 2** | **address-family** { **ipv4** \| **ipv6** } [ **unicast** ]<br>**Example:**<br>The following is an example for ipv4 address family:<br>`RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast` | Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. |
| **Step 3** | **segment-routing prefix-sid-map advertise-local**<br>**Example:**<br>`RP/0/RSP0/CPU0:router(config-isis-af)# segment-routing prefix-sid-map advertise-local` | Configures IS-IS to advertise locally configured prefix-SID mappings. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br><br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify IS-IS prefix-SID mapping advertisement and TLV.

```
RP/0/RSP0/CPU0:router# show isis database verbose

<...removed...>

 SID Binding:  10.1.1.1/32 F:0 M:0 S:0 D:0 A:0 Weight:0 Range:200
    SID: Start:10, Algorithm:0, R:0 N:0 P:0 E:0 V:0 L:0
 SID Binding:  20.1.1.0/24 F:0 M:0 S:0 D:0 A:0 Weight:0 Range:300
    SID: Start:400, Algorithm:0, R:0 N:0 P:0 E:0 V:0 L:0
```

# Configure Mapping Advertisement for OSPF

### SUMMARY STEPS

1. **router ospf** *process-name*
2. **segment-routing prefix-sid-map advertise-local**
3. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **router ospf** *process-name*<br><br>**Example:**<br><br>`RP/0/RSP0/CPU0:router(config)# router ospf 1` | Enables OSPF routing for the specified routing instance, and places the router in router configuration mode. |
| **Step 2** | **segment-routing prefix-sid-map advertise-local**<br><br>**Example:**<br><br>`RP/0/RSP0/CPU0:router(config-ospf)# segment-routing prefix-sid-map advertise-local` | Configures OSPF to advertise locally configured prefix-SID mappings. |
| **Step 3** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session. |

| | Command or Action | Purpose |
|---|---|---|
| | | **end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

Verify OSP prefix-SID mapping advertisement and TLV.

```
RP/0/RSP0/CPU0:router# show ospf database opaque-area

<...removed...>

    Extended Prefix Range TLV: Length: 24
      AF        : 0
      Prefix    : 10.1.1.1/32
      Range Size: 200
      Flags     : 0x0

      SID sub-TLV: Length: 8
        Flags     : 0x60
        MTID      : 0
        Algo      : 0
        SID Index : 10
```

# Enable Mapping Client

By default, mapping client functionality is enabled.

You can disable the mapping client functionality by using the **segment-routing prefix-sid-map receive disable** command.

You can re-enable the mapping client functionality by using the **segment-routing prefix-sid-map receive** command.

The following example shows how to enable the mapping client for IS-IS:

```
RP/0/RSP0/CPU0:router(config)# router isis 1
RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-isis-af)# segment-routing prefix-sid-map receive
```

The following example shows how to enable the mapping client for OSPF:

```
RP/0/RSP0/CPU0:router(config)# router ospf 1
RP/0/RSP0/CPU0:router(config-ospf)# segment-routing prefix-sid-map receive
```

CHAPTER 12

# Using Segment Routing Traffic Matrix

This module provides information about the Segment Routing Traffic Matrix (SR-TM) and the Traffic Collector process, and describes how to configure the TM border and the Traffic Collector and to display traffic information.



## Segment Routing Traffic Matrix

A network's traffic matrix is a description, measure, or estimation of the aggregated traffic flows that enter, traverse, and leave a network.

The Segment Routing Traffic Matrix (SR-TM) is designed to help users understand traffic patterns on a router. The Traffic Matrix border divides the network into two parts: internal (interfaces that are inside the border) and external (interfaces that are outside the border). By default, all interfaces are internal. You can configure an interface as external.

## Traffic Collector Process

The Traffic Collector collects packet and byte statistics from router components such as prefix counters, tunnel counters, and the TM counter. The TM counter increments when traffic that comes from an external interface to the network is destined for a segment routing prefix-SID. The Traffic Collector keeps histories of the statistics and makes them persistent across process restarts, failovers, and ISSU. Histories are retained for a configurable length of time.

### Pcounters

A Pcounter is a packet and byte pair of counters. There is one Pcounter per tunnel. There are two Pcounters per prefix-SID:

- Base Pcounter – any packet that is switched on the prefix-SID forwarding information base (FIB) entry
- TM Pcounter – any packet from an external interface and switched on the prefix-SID FIB entry

The Traffic Collector periodically collects the Base Pcounters and TM Pcounters of all prefix-SIDs, and the Pcounters of all tunnel interfaces.

For each Pcounter, the Traffic Collector calculates the number of packets and bytes that have been forwarded during the last interval. The Traffic Collector keeps a history of the per-interval statistics for each of the Pcounters. Each entry in the history contains:

- The start and end time of the interval
- The number of packets forwarded during the interval
- The number of bytes forwarded during the interval

### Feature Support and Limitations

- Pcounters for IPv4 SR Prefix SIDs are supported.
- Pcounters for IPv6 SR Prefix SIDs are not supported.
- TM Pcounters increment for incoming SR-labeled and IP traffic destined for an SR Prefix SID.
- External interface support can be enabled on all Ethernet interfaces except Management, Bundle, and sub interfaces. Tunnels may not be set as external interfaces.
- Default VRF is supported. Non-default VRF is not supported.

# Configuring Traffic Collector

Perform these tasks to configure the traffic collector.

### SUMMARY STEPS

1. **configure**
2. **traffic-collector**
3. **statistics collection-interval** *value*
4. **statistics history-size** *value*
5. **statistics history-timeout** *value*
6. **interface** *type l3-interface-address*
7. Use the **commit** or **end** command.

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure**<br>**Example:**<br>`RP/0/RSP0/CPU0:router# configure` | Enters global configuration mode. |
| **Step 2** | **traffic-collector**<br>**Example:** | Enables traffic collector and places the router in traffic collector configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| | `RP/0/RP0/CPU0:router(config)# traffic-collector` | |
| **Step 3** | **statistics collection-interval** *value*<br>**Example:**<br>`RP/0/RP0/CPU0:router(config-tc)# statistics collection-interval 5` | (Optional) Sets the frequency that the traffic collector collects and posts data, in minutes. Valid values are 1, 2, 3, 4, 5, 6, 10, 12,15, 20, 30, and 60. The default interval is 1. |
| **Step 4** | **statistics history-size** *value*<br>**Example:**<br>`RP/0/RP0/CPU0:router(config-tc)# statistics history-size 10` | (Optional) Specifies the number of entries kept in the history database. Valid values are from 1 to 10. The default is 5.<br>**Note** The number of entries affects how the average packet and average byte rates are calculated. The rates are calculated over the range of the histories and are not averages based in real time. |
| **Step 5** | **statistics history-timeout** *value*<br>**Example:**<br>`RP/0/RP0/CPU0:router(config-tc)# statistics history-timeout 24` | (Optional) When a prefix SID or a tunnel-te interface is deleted, the history-timeout sets the length of time, in hours, that the prefix SID and tunnel statistics are retained in the history before they are removed. The minimum is one hour; the maximum is 720 hours. The default is 48.<br>**Note** Enter 0 to disable the history timeout. (No history is retained.) |
| **Step 6** | **interface** *type l3-interface-address*<br>**Example:**<br>`RP/0/RP0/CPU0:router(config-tc)# interface TenGigE 0/1/0/3` | Identifies interfaces that handle external traffic. Only L3 interfaces are supported for external traffic. |
| **Step 7** | Use the **commit** or **end** command. | **commit** —Saves the configuration changes and remains within the configuration session.<br>**end** —Prompts user to take one of these actions:<br>• **Yes** — Saves configuration changes and exits the configuration session.<br>• **No** —Exits the configuration session without committing the configuration changes.<br>• **Cancel** —Remains in the configuration session, without committing the configuration changes. |

This completes the configuration for the traffic collector.

# Displaying Traffic Information

The following show commands display information about the interfaces and tunnels:

**Note** For detailed information about the command syntax for the following **show** commands, see the *Segment Routing Command Reference Guide*.

- Display the configured external interfaces:

```
RP/0/RSP0/CPU0:router# show traffic-collector external-interface
  Interface             Status
  --------------------  ----------------
  Te0/1/0/3             Enabled
  Te0/1/0/4             Enabled
```

- Display the counter history database for a prefix-SID:

```
RP/0/RSP0/CPU0:router# show traffic-collector ipv4 counters prefix 1.1.1.10/32 detail
Prefix: 1.1.1.10/32  Label: 16010 State: Active
Base:
    Average over the last 5 collection intervals:
        Packet rate: 9496937 pps, Byte rate: 9363979882 Bps

    History of counters:
        23:01 - 23:02: Packets 9379529, Bytes: 9248215594
        23:00 - 23:01: Packets 9687124, Bytes: 9551504264
        22:59 - 23:00: Packets 9539200, Bytes: 9405651200
        22:58 - 22:59: Packets 9845278, Bytes: 9707444108
        22:57 - 22:58: Packets 9033554, Bytes: 8907084244
TM Counters:
    Average over the last 5 collection intervals:
        Packet rate: 9528754 pps, Byte rate: 9357236821 Bps

    History of counters:
        23:01 - 23:02: Packets 9400815, Bytes: 9231600330
        23:00 - 23:01: Packets 9699455, Bytes: 9524864810
        22:59 - 23:00: Packets 9579889, Bytes: 9407450998
        22:58 - 22:59: Packets 9911734, Bytes: 9733322788
        22:57 - 22:58: Packets 9051879, Bytes: 8888945178
```

This output shows the average Pcounter (packets, bytes), the Pcounter history, and the collection interval of the Base and TM for the specified prefix-SID.

- Display the counter history database for a policy:

```
RP/0/RSP0/CPU0:router# show traffic-collector counters tunnels srte_c_12_ep_6.6.6.2
detail
Tunnel: srte_c_12_ep_6.6.6.2  State: Active
    Average over the last 5 collection intervals:
        Packet rate: 9694434 pps, Byte rate: 9597489858 Bps

    History of counters:
        23:14 - 23:15: Packets 9870522 , Bytes: 9771816780
```

```
23:13 - 23:14: Packets 9553048 , Bytes: 9457517520
23:12 - 23:13: Packets 9647265 , Bytes: 9550792350
23:11 - 23:12: Packets 9756654 , Bytes: 9659087460
23:10 - 23:11: Packets 9694434 , Bytes: 9548235180
```

This output shows the average Pcounter (packets, bytes), the Pcounter history, and the collection interval for the policy.

CHAPTER 13

# Using Segment Routing OAM

Segment Routing Operations, Administration, and Maintenance (OAM) helps service providers to monitor label-switched paths (LSPs) and quickly isolate forwarding problems to assist with fault detection and troubleshooting in the network. The Segment Routing OAM feature provides support for IGP prefix SIDs, and Nil-FEC (forwarding equivalence classes) LSP Ping and Traceroute functionality.



## MPLS Ping and Traceroute for BGP and IGP Prefix-SID

MPLS Ping and Traceroute operations for Prefix SID are supported for various IGP scenarios, for example:

- Within an IS-IS level or OSPF area
- Across IS-IS levels or OSPF areas
- Route redistribution from IS-IS to OSPF and from OSPF to IS-IS
- Anycast Prefix SID

The MPLS LSP Ping feature is used to check the connectivity between ingress Label Switch Routers (LSRs) and egress LSRs along an LSP. MPLS LSP ping uses MPLS echo request and reply messages, similar to Internet Control Message Protocol (ICMP) echo request and reply messages, to validate an LSP. The destination IP address of the MPLS echo request packet is different from the address used to select the label stack. The destination IP address is defined as a 127.x.y.z/8 address and it prevents the IP packet from being IP switched to its destination, if the LSP is broken.

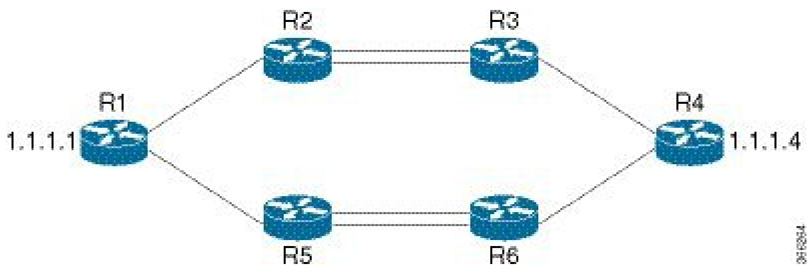
The MPLS LSP Traceroute feature is used to isolate the failure point of an LSP. It is used for hop-by-hop fault localization and path tracing. The MPLS LSP Traceroute feature relies on the expiration of the Time to Live (TTL) value of the packet that carries the echo request. When the MPLS echo request message hits a transit node, it checks the TTL value and if it is expired, the packet is passed to the control plane, else the message is forwarded. If the echo message is passed to the control plane, a reply message is generated based on the contents of the request message.

The MPLS LSP Tree Trace (traceroute multipath) operation is also supported for IGP Prefix SID. MPLS LSP Tree Trace provides the means to discover all possible equal-cost multipath (ECMP) routing paths of an LSP to reach a destination Prefix SID. It uses multipath data encoded in echo request packets to query for the load-balancing information that may allow the originator to exercise each ECMP. When the packet TTL expires at the responding node, the node returns the list of downstream paths, as well as the multipath information that can lead the operator to exercise each path in the MPLS echo reply. This operation is performed repeatedly for each hop of each path with increasing TTL values until all ECMP are discovered and validated.

MPLS echo request packets carry Target FEC Stack sub-TLVs. The Target FEC sub-TLVs are used by the responder for FEC validation. The IGP IPv4 prefix sub-TLV has been added to the Target FEC Stack sub-TLV. The IGP IPv4 prefix sub-TLV contains the prefix SID, the prefix length, and the protocol (IS-IS or OSPF).

# Examples: MPLS Ping, Traceroute, and Tree Trace for Prefix-SID

These examples use the following topology:




### MPLS Ping for Prefix-SID

```
RP/0/RSP0/CPU0:router-arizona# ping mpls ipv4 1.1.1.4/32
Thu Dec 17 01:01:42.301 PST

Sending 5, 100-byte MPLS Echos to 1.1.1.4,
      timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.

!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
```

### MPLS Traceroute for Prefix-SID

```
RP/0/RSP0/CPU0:router-arizona# traceroute mpls ipv4 1.1.1.4/32
Thu Dec 17 14:45:05.563 PST

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
```

```
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.

  0 12.12.12.1 MRU 4470 [Labels: 16004 Exp: 0]
L 1 12.12.12.2 MRU 4470 [Labels: 16004 Exp: 0] 3 ms
L 2 23.23.23.3 MRU 4470 [Labels: implicit-null Exp: 0] 3 ms
! 3 34.34.34.4 11 ms
```

### MPLS Tree Trace for Prefix-SID

```
RP/0/RSP0/CPU0:router-arizona# traceroute mpls multipath ipv4 1.1.1.4/32
Thu Dec 17 14:55:46.549 PST

Starting LSP Path Discovery for 1.1.1.4/32

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no rx label,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.

LL!
Path 0 found,
 output interface TenGigE0/0/0/0 nexthop 12.12.12.2  source 12.12.12.1 destination 127.0.0.0
 L!
Path 1 found,
 output interface TenGigE0/0/0/0 nexthop 12.12.12.2  source 12.12.12.1 destination 127.0.0.2
 LL!
Path 2 found,
 output interface TenGigE0/0/0/1 nexthop 15.15.15.5  source 15.15.15.1 destination 127.0.0.1
 L!
Path 3 found,
 output interface TenGigE0/0/0/1 nexthop 15.15.15.5  source 15.15.15.1 destination 127.0.0.0

Paths (found/broken/unexplored) (4/0/0)
 Echo Request (sent/fail) (10/0)
 Echo Reply (received/timeout) (10/0)
 Total Time Elapsed 53 ms
```

# MPLS LSP Ping and Traceroute Nil FEC Target

The Nil-FEC LSP ping and traceroute operations are extensions of regular MPLS ping and traceroute.

Nil-FEC LSP Ping/Traceroute functionality supports segment routing and MPLS Static. It also acts as an additional diagnostic tool for all other LSP types. This feature allows operators to provide the ability to freely test any label stack by allowing them to specify the following:

- label stack
- outgoing interface
- nexthop address

In the case of segment routing, each segment nodal label and adjacency label along the routing path is put into the label stack of an echo request message from the initiator Label Switch Router (LSR); MPLS data plane forwards this packet to the label stack target, and the label stack target sends the echo message back.

The following table shows the syntax for the ping and traceroute commands.

*Table 3: LSP Ping and Traceroute Nil FEC Commands*

| Command Syntax |
| --- |
| **ping mpls nil-fec labels** {*label*[**,***label*]} [**output** {**interface** *tx-interface*} [**nexthop** *nexthop-ip-addr*]] |
| **traceroute mpls nil-fec labels** {*label*[**,***label*]} [**output** {**interface** *tx-interface*} [**nexthop** *nexthop-ip-addr*]] |

# Examples: LSP Ping and Traceroute for Nil_FEC Target

These examples use the following topology:

```
Node loopback IP address: 172.18.1.3   172.18.1.4   172.18.1.5   172.18.1.7
Node label:                            16004        16005        16007
Nodes:                    Arizona ---- Utah ------- Wyoming ---- Texas

Interface:            GigabitEthernet0/2/0/1   GigabitEthernet0/2/0/1
Interface IP address:         10.1.1.3              10.1.1.4
```

```
RP/0/RSP0/CPU0:router-utah# show mpls forwarding

Tue Jul  5 13:44:31.999 EDT
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16004  Pop         No ID              Gi0/2/0/1    10.1.1.4        1392
       Pop         No ID              Gi0/2/0/2    10.1.2.2        0
16005  16005       No ID              Gi0/2/0/0    10.1.1.4        0
       16005       No ID              Gi0/2/0/1    10.1.2.2        0
16007  16007       No ID              Gi0/2/0/0    10.1.1.4        4752
       16007       No ID              Gi0/2/0/1    10.1.2.2        0
24000  Pop         SR Adj (idx 0)     Gi0/2/0/0    10.1.1.4        0
24001  Pop         SR Adj (idx 2)     Gi0/2/0/0    10.1.1.4        0
24002  Pop         SR Adj (idx 0)     Gi0/2/0/1    10.1.2.2        0
24003  Pop         SR Adj (idx 2)     Gi0/2/0/1    10.1.2.2        0
24004  Pop         No ID              tt10         point2point     0
24005  Pop         No ID              tt11         point2point     0
24006  Pop         No ID              tt12         point2point     0
24007  Pop         No ID              tt13         point2point     0
24008  Pop         No ID              tt30         point2point     0
```

### Ping Nil FEC Target

```
RP/0/RSP0/CPU0:router-arizona# ping mpls nil-fec labels 16005,16007 output interface
GigabitEthernet 0/2/0/1 nexthop 10.1.1.4 repeat 1
Sending 1, 72-byte MPLS Echos with Nil FEC labels 16005,16007,
     timeout is 2 seconds, send interval is 0 msec:
```

```
Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'l' - Label switched with FEC change, 'd' - see DDMAP for return code,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
!
Success rate is 100 percent (1/1), round-trip min/avg/max = 1/1/1 ms
 Total Time Elapsed 0 ms
```

### Traceroute Nil FEC Target

```
RP/0/RSP0/CPU0:router-arizona# traceroute mpls nil-fec labels 16005,16007 output interface
 GigabitEthernet 0/2/0/1 nexthop 10.1.1.4
Tracing MPLS Label Switched Path with Nil FEC labels 16005,16007, timeout is 2 seconds

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'l' - Label switched with FEC change, 'd' - see DDMAP for return code,
  'X' - unknown return code, 'x' - return code 0

Type escape sequence to abort.
  0 10.1.1.3 MRU 1500 [Labels: 16005/16007/explicit-null Exp: 0/0/0]
L 1 10.1.1.4 MRU 1500 [Labels: implicit-null/16007/explicit-null Exp: 0/0/0] 1 ms
L 2 10.1.1.5 MRU 1500 [Labels: implicit-null/explicit-null Exp: 0/0] 1 ms
! 3 10.1.1.7 1 ms
```