# CONTENTS

## Preface
- Preface vii
  - Changes to This Document vii
  - Obtaining Documentation and Submitting a Service Request vii

## Chapter 1
- New and Changed Information for Segment Routing Features 1
  - New and Changed Information 1

## Chapter 2
- About Segment Routing 3
  - Scope 3
  - Need 4
  - Benefits 4
  - Workflow for Deploying Segment Routing 5

## Chapter 3
- Configure Segment Routing Global Block 7
  - About the Segment Routing Global Block 7
  - Setup a Non-Default Segment Routing Global Block Range 8

## Chapter 4
- Configure Segment Routing for IS-IS Protocol 11
  - Enabling Segment Routing for IS-IS Protocol 11
  - Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface 13
  - IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability 16
    - Prefix Attribute Flags 16
    - IPv4 and IPv6 Source Router ID 17
    - Configuring Prefix Attribute N-flag-clear 18
  - IS-IS Multi-Domain Prefix SID and Domain Stitching: Example 19
    - Configure IS-IS Multi-Domain Prefix SID 19
    - Configure Common Router ID 20
### CHAPTER 5

**Configure Segment Routing for OSPF Protocol**  
- Enabling Segment Routing for OSPF Protocol  
- Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface

### CHAPTER 6

**Configure Segment Routing for BGP**  
- Segment Routing for BGP  
- Configure BGP Prefix Segment Identifiers  
- Configure Segment Routing Egress Peer Engineering  
- Configure BGP Link-State  
- Example: Configuring SR-EPE and BGP-LS

### CHAPTER 7

**Configure SR-TE Policies**  
- About SR-TE Policies  
- How to Configure SR-TE Policies  
  - Configure Local Dynamic SR-TE Policy  
  - Configure Explicit SR-TE Policy  
- Steering Traffic into an SR-TE Policy  
  - Configure Static Routes  
  - Configure Autoroute Announce  
  - Configure Autoroute Destination  
- BGP SR-TE  
  - Configure Dynamic BGP SR-TE  
  - Configure Explicit BGP SR-TE  
- Using Binding Segments  
  - Stitching SR-TE Polices Using Binding SID: Example  
- Configure Seamless Bidirectional Forwarding Detection  
  - Configure the SBFD Reflector  
  - Configure the SBFD Initiator  
  - Enable Line Cards to Host BFD Sessions  
  - Map a Destination Address to a Remote Discriminator  
  - Enable SBFD on the SR-TE Policy

### CHAPTER 8

**Configure IOS XR Traffic Controller (XTC)**  
- Distribute IS-IS Link-State Data  
- Configure Segment Routing for OSPF Protocol  
- Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface  
- Configure Segment Routing for BGP  
- Segment Routing for BGP  
- Configure BGP Prefix Segment Identifiers  
- Configure Segment Routing Egress Peer Engineering  
- Configure BGP Link-State  
- Example: Configuring SR-EPE and BGP-LS  
- Configure SR-TE Policies  
- About SR-TE Policies  
- How to Configure SR-TE Policies  
  - Configure Local Dynamic SR-TE Policy  
  - Configure Explicit SR-TE Policy  
- Steering Traffic into an SR-TE Policy  
  - Configure Static Routes  
  - Configure Autoroute Announce  
  - Configure Autoroute Destination  
- BGP SR-TE  
  - Configure Dynamic BGP SR-TE  
  - Configure Explicit BGP SR-TE  
- Using Binding Segments  
  - Stitching SR-TE Polices Using Binding SID: Example  
- Configure Seamless Bidirectional Forwarding Detection  
  - Configure the SBFD Reflector  
  - Configure the SBFD Initiator  
  - Enable Line Cards to Host BFD Sessions  
  - Map a Destination Address to a Remote Discriminator  
  - Enable SBFD on the SR-TE Policy  
- Configure IOS XR Traffic Controller (XTC)
Contents

CHAPTER 9  Configure Topology-Independent Loop-Free Alternate (TI-LFA) 91
  Configuring TI-LFA for IS-IS 91
  Configuring TI-LFA for OSPF 93
  TI-LFA Node and SRLG Protection: Examples 95
  Configuring and Verifying TI-LFA: Example 96

CHAPTER 10  Configure Segment Routing Microloop Avoidance 101
  About Segment Routing Microloop Avoidance 101
  Configure Segment Routing Microloop Avoidance for IS-IS 101
  Configure Segment Routing Microloop Avoidance for OSPF 102

CHAPTER 11  Configure Segment Routing Mapping Server 105
  Segment Routing Mapping Server 105
    Segment Routing Mapping Server Restrictions 106
  Segment Routing and LDP Interoperability 106
    Example: Segment Routing LDP Interoperability 106
  Configuring Mapping Server 108
  Enable Mapping Advertisement 110
    Configure Mapping Advertisement for IS-IS 110
    Configure Mapping Advertisement for OSPF 111
  Enable Mapping Client 112

CHAPTER 12  Collecting Traffic Statistics 113
  Traffic Collector Process 113
CHAPTER 13

Using Segment Routing OAM 117

MPLS Ping and Traceroute for BGP and IGP Prefix-SID 117
Examples: MPLS Ping, Traceroute, and Tree Trace for Prefix-SID 118
MPLS LSP Ping and Traceroute Nil FEC Target 119
Examples: LSP Ping and Traceroute for Nil_FEC Target 120
Preface

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

- Changes to This Document, page vii
- Obtaining Documentation and Submitting a Service Request, page vii

Changes to This Document

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 2017</td>
<td>Initial release of this document</td>
</tr>
</tbody>
</table>

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. RSS feeds are a free service.
# 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, page 1

## Table 1: New and Changed Features

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

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

- Scope, page 3
- Need, page 4
- Benefits, page 4
- Workflow for Deploying Segment Routing, page 5

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 on the IGP
3. Configure Segment Routing on the BGP
4. Configure the SR-TE Policy
5. Configure TI-LFA
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, page 7
- Setup a Non-Default Segment Routing Global Block Range, page 8

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. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> [router {isis instance-id</td>
<td>ospf process_name} ]</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router isis 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> segment-routing global-block starting_value ending_value</td>
<td>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.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis)# segment-routing global-block 18000 19999</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
Verify the SRGB configuration:

RP/0/RSP0/CPU0:router# show mpls label table detail
Table Label  Owner                  State Rewrite
---------- -------------- --------------
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 SID s and enable segment routing.
Setup a Non-Default Segment Routing Global Block Range
Configure Segment Routing for IS-IS Protocol


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

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, page 11
- Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface, page 13
- IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability, page 16
- IS-IS Multi-Domain Prefix SID and Domain Stitching: Example, page 19

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. segment-routing mpls
8. exit
9. mpls traffic-eng
10. commit

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config)# router isis isp</td>
<td>You can change the level of routing to be performed by a particular routing instance by using the <code>is-type</code> router configuration command.</td>
</tr>
<tr>
<td>Step 3 address-family { ipv4</td>
<td>ipv6 } [ unicast ]</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-isis)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Step 4 metric-style wide [ level { 1</td>
<td>2 }]</td>
</tr>
<tr>
<td><strong>Example:</strong> RP/0/RSP0/CPU0:router(config-isis-af)# metric-style wide level 1</td>
<td></td>
</tr>
</tbody>
</table>
### Configure Segment Routing for IS-IS Protocol

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

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. The prefix segment steers the traffic along the shortest path to its

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong></td>
<td>mpls traffic-eng <strong>level</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng level-2-only</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>mpls traffic-eng router-id <strong>interface</strong>&lt;br&gt;Example:&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis-af)# mpls traffic-eng router-id Loopback0</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>segment-routing mpls&lt;br&gt;Example:&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis-af)# segment-routing mpls</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>exit&lt;br&gt;Example:&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis-af)# exit&lt;br&gt;RP/0/RSP0/CPU0:router(config-isis)# exit</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>mpls traffic-eng&lt;br&gt;Example:&lt;br&gt;RP/0/RSP0/CPU0:router(config)# mpls traffic-eng</td>
</tr>
<tr>
<td><strong>Step 10</strong></td>
<td>commit</td>
</tr>
</tbody>
</table>

---

**What to Do Next**

Configure the prefix SID.
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.

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.

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

Note

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. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config) #</td>
<td></td>
</tr>
<tr>
<td>router isis 1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> interface Loopback</td>
<td>Specifies the loopback interface and instance.</td>
</tr>
<tr>
<td>instance</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis)</td>
<td></td>
</tr>
<tr>
<td># interface Loopback0</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> address-family</td>
<td>Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode.</td>
</tr>
<tr>
<td>{ipv4</td>
<td>ipv6 } [ unicast ]</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Configure Segment Routing for IS-IS Protocol

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following is an example for ipv4 address family:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> prefix-sid [strict-spf] {index SID-index</td>
<td>absolute SID-value } [n-flag-clear ] [ explicit-null ]</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# prefix-sid index 1001</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# prefix-sid strict-spf index 101</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# prefix-sid absolute 17001</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

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: 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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X (External Prefix Flag)</td>
<td>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.</td>
</tr>
<tr>
<td>R (Re-advertisement Flag)</td>
<td>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). 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).</td>
</tr>
</tbody>
</table>
For prefixes that are propagated from another level:
1 Copy the N-flag from the prefix attribute sub-TLV, if present in the source level.
2 Copy the N-flag from the prefix-SID sub-TLV, if present in the source level.
3 Otherwise, set to 0.

For connected prefixes:
1 Set to 0 if `isis prefix-attributes n-flag-clear` is configured (see Configuring Prefix Attribute N-flag-clear).
2 Set to 0 if `prefix-SID n-flag-clear` is configured (see Configuring a Prefix-SID on the IS-IS Enabled Loopback Interface).
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.

**Note** If the flag is set and the prefix length is not a host prefix, then the flag must be ignored.

---

**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**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| IPv4 and IPv6 Source Router ID | **Type**: 11  
|                             | **Length**: 4  
|                             | **Value**: IPv4 Router ID of the source of the prefix advertisement     |
| IPv6 Source Router ID       | **Type**: 12  
|                             | **Length**: 16  
|                             | **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. interface Loopback  *instance*
3. isis prefix-attributes n-flag-clear [Level-1 | Level-2]
4. commit

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> interface Loopback  <em>instance</em></td>
<td>Specifies the loopback interface and instance.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>interface Loopback0</td>
</tr>
<tr>
<td><strong>Step 3</strong> isis prefix-attributes n-flag-clear [Level-1</td>
<td>Level-2]</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-if)#</td>
<td></td>
</tr>
<tr>
<td></td>
<td>isis prefix-attributes n-flag-clear</td>
</tr>
<tr>
<td><strong>Step 4</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>

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
    NLSPID: 0x8e
    MT: Standard (IPv4 Unicast)
    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
<...>
```
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 IOS XR Traffic Controller (XTC) (see Configure IOS XR Traffic Controller (XTC)).

**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
```
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
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  
  router isis Domain2
  address-family ipv4 unicast
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  ```

- **Example: Border Node 9**
  
  ```
  router isis Domain1
  address-family ipv4 unicast
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  
  router isis Domain2
  address-family ipv4 unicast
  mpls traffic-eng level-2-only
  mpls traffic-eng 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 bgp-ls level 2
  ```

- **Example: Node 14**
  
  ```
  router isis Domain2
  distribute bgp-ls level 2
  ```
Nodes 13 and 14 each report 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.
Distribute IS-IS Link-State Data
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

Enabling Segment Routing for OSPF Protocol, page 23
Configuring a Prefix-SID on the OSPF-Enabled Loopback Interface, page 25

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.
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. segment-routing mpls
4. area 0
5. mpls traffic-eng area
6. mpls traffic-eng router-id interface
7. segment-routing mpls
8. exit
9. mpls traffic-eng
10. commit

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td>Enables OSPF routing for the specified routing process and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2</td>
<td><code>router ospf process-name</code></td>
<td>Enables segment routing using the MPLS data plane on the routing process and all areas and interfaces in the routing process. Enables segment routing forwarding on all interfaces in the routing process and installs the SIDs received by OSPF in the forwarding table.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>segment-routing mpls</td>
<td>Enters area configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# segment-routing mpls</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>area 0</td>
<td>Enables IGP traffic engineering functionality.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 0</td>
<td></td>
</tr>
</tbody>
</table>
### Configure Segment Routing for OSPF Protocol

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

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. 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.

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.

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 6</td>
<td>mpls traffic-eng router-id interface</td>
<td>Sets the traffic engineering loopback interface.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# mpls traffic-eng router-id Loopback0</td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>segment-routing mpls</td>
<td>(Optional) Enables segment routing using the MPLS data plane on the area and all interfaces in the area. Enables segment routing forwarding on all interfaces in the area and installs the SIDs received by OSPF in the forwarding table.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# segment-routing mpls</td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td>exit</td>
<td>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 SR-TE head-end requires the TED to calculate and validate the path of the SR-TE policy.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# exit</td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>mpls traffic-eng</td>
<td>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 SR-TE head-end requires the TED to calculate and validate the path of the SR-TE policy.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# mpls traffic-eng</td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

### What to Do Next
Configure the prefix SID.

---

Configure Segment Routing for OSPF Protocol

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

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. 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.

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` `{ index *SID-index* | absolute *SID-value* } [n-flag-clear] [ explicit-null ]`
6. commit

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables OSPF routing for the specified routing process, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> <code>router ospf</code> <em>process-name</em></td>
<td>Enables OSPF routing for the specified routing process, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config)# router ospf 1</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> <code>area</code> <em>value</em></td>
<td>Enters area configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-ospf)# area 0</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> <code>interface Loopback</code> <em>interface-instance</em></td>
<td>Specifies the loopback interface and instance.</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface Loopback0 passive</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> <code>prefix-sid</code> `{ index <em>SID-index</em></td>
<td>absolute <em>SID-value</em> } [n-flag-clear] [ explicit-null ]`</td>
</tr>
<tr>
<td><strong>Example:</strong> <code>RP/0/RSP0/CPU0:router(config-ospf-ar)# prefix-sid index 1001</code></td>
<td></td>
</tr>
<tr>
<td><code>RP/0/RSP0/CPU0:router(config-ospf-ar)# prefix-sid absolute 17001</code></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> commit</td>
<td></td>
</tr>
</tbody>
</table>
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
    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 the SR-TE policy.
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.

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, page 29
- Configure BGP Prefix Segment Identifiers, page 30
- Configure Segment Routing Egress Peer Engineering, page 31
- Configure BGP Link-State, page 32
- Example: Configuring SR-EPE and BGP-LS, page 33

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)# 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):
```
Configure 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.

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.

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**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>router bgp as-number</code></td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# <code>router bgp 1</code></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>neighbor ip-address</code></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-bgp)# <code>neighbor 192.168.1.3</code></td>
<td></td>
</tr>
</tbody>
</table>
### 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:

```plaintext
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 Extensions**

A given BGP node may have connections to multiple, independent routing domains; IGP link state distribution into BGP has been added for both OSPF and ISIS protocols to enable that node to pass this information, in a similar fashion, on to applications that desire to build paths spanning or including these multiple domains.
To distribute ISIS link-state data using BGP LS, use the `distribute bgp-ls` 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 bgp-ls instance-id 32 level 2 throttle 5
```

To distribute OSPFv2 and OSPFv3 link-state data using BGP LS, use the `distribute bgp-ls` 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 bgp-ls 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
ergess-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
ergess-engineering
RP/0/RSP0/CPU0:router_C(config-bgp-nbr)# description to D
RP/0/RSP0/CPU0:router_C(config-bgp-nbr-af)# 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
```
Step 2 Configure node C to advertise peer node SIDs to the controller using BGP-LS.

Example:

```bash
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 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:

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

Step 4 Verify the configuration.

Example:

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

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

Example:

```bash
RP/0/RSP0/CPU0:router_C# show mpls forwarding labels 24002 24003
```

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, page 37
- How to Configure SR-TE Policies, page 38
- Steering Traffic into an SR-TE Policy, page 41
- BGP SR-TE, page 46
- Using Binding Segments, page 52
- Configure Seamless Bidirectional Forwarding Detection, page 55

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, on page 38
- Configure Explicit SR-TE Policy, on page 39

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>interface tunnel-te tunnel-id</td>
<td>Configures the tunnel interface.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RSP0/CPU0:router(config)# interface tunnel-te22</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>ipv4 unnumbered type interface-path-id</td>
<td>Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>destination ip-address</td>
<td>Assigns a destination address on the new tunnel.</td>
</tr>
<tr>
<td></td>
<td>Example: RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2</td>
<td></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enters a name for the explicit path and enters the explicit path configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong> explicit-path name <em>path-name</em></td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# explicit-path</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>name rir6_exp</td>
<td></td>
</tr>
</tbody>
</table>

**Step 3**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>index index {next-address ip-address</td>
<td>next-label label}</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.
- Each entry must have a unique index.
- 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.

**Example:**

RP/0/RSP0/CPU0:router(config-expl-path)#
index 1 next-label 16001
RP/0/RSP0/CPU0:router(config-expl-path)#
index 2 next-label 16006

**Step 4**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>exit</td>
<td></td>
</tr>
</tbody>
</table>

**Step 5**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface tunnel-te tunnel-id</td>
<td>Configures the tunnel interface.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

RP/0/RSP0/CPU0:router(config)# interface tunnel-te22

**Step 6**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipv4 unnumbered type interface-path-id</td>
<td>Assigns a source address so that forwarding can be performed on the new tunnel. Loopback is commonly used as the interface type.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0

**Step 7**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination ip-address [verbatim]</td>
<td>Assigns a destination address on the new tunnel. 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.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2

**Step 8**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>path-option preference-priority explicit name path-name segment-routing</td>
<td>Specifies the explicit path name and assigns the path ID.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

RP/0/RSP0/CPU0:router(config-if)# path-option 1 explicit name rir6_exp segment-routing

**Step 9**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>interface tunnel-te tunnel-id</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config)# interface tunnel-te22</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>ipv4 unnumbered type interface-path-id</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>autoroute announce</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# autoroute announce</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>destination ip-address</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td><strong>path-option preference-priority dynamic segment-routing</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# path-option 1 dynamic segment-routing</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td><strong>path-protection</strong>&lt;br&gt;<strong>Example:</strong>&lt;br&gt;RP/0/RSP0/CPU0:router(config-if)# path-protection</td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td><strong>commit</strong></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>interface tunnel-te tunnel-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# interface tunnel-te22</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>ipv4 unnumbered type interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# ipv4 unnumbered loopback0</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>autoroute destination destination-ip-address</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.1</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.2 (the default route)</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.3</td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-if)# autoroute destination 192.168.0.4</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>destination ip-address</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-if)# destination 192.168.0.2</td>
</tr>
<tr>
<td>Step 6</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>path-option preference-priority dynamic segment-routing</td>
</tr>
</tbody>
</table>

### 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.

BGP SR-TE uses routing policies to instruct the router to inspect routes, filter them, and potentially modify their attributes as they are accepted from a peer, advertised to a peer, or redistributed from one routing protocol to another. Routing policy language (RPL) functionality has been enhanced to provide for an attribute-set parameter that is set on an individual BGP-prefix basis (for use with existing matches). Using existing routing policy language (RPL) configuration, BGP provides a way to tag prefixes using communities and associate each community with an attribute-set or profile. The profile is locally configured to define certain TE parameters, such as latency, bandwidth, and explicit paths. Once BGP prefixes are mapped to a profile, TE dynamically creates a tunnel using parameters defined in the attribute-set.

**Note**

For more information on routing policies and routing policy language (RPL), refer to the "Implementing Routing Policy" chapter in the Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide.

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

### Dynamic BGP SR-TE

Dynamic BGP SR-TE uses paths that are computed by the head-end or by IOS XR Traffic Controller (XTC). If a BGP receiver node is configured with an attribute-set, a TE tunnel is created by BGP when a routing policy associates the received route with the attribute-set. The routing policy can specify a path requirement (for example, a low-latency path or minimum-cost path).

Dynamic SR-TE can be distributed, where the head-end computes the path locally and sets up the tunnel along the path based on the configured attribute-set, or centralized, where the head-end requests that XTC computes the path.

**Note**

"Centralized" indicates the capability of the XTC, and not its location in the network.
You can use XTC when the local computation is not possible (for example, if more information is needed than is available on the head-end). For information on XTC, see the Configure IOS XR Traffic Controller (XTC) chapter.

**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 an 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.

**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

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td><code>route-policy route-policy-name</code></td>
<td>Creates a route policy and enters route policy configuration mode, where you can define the route policy.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config)# route-policy BGP_TE</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-rpl)# if community matches-every (100:1) then</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-rpl-if)# set mpls traffic-eng attribute-set low_lat</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-rpl-if)# endif</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-rpl)# pass</code></td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td><code>end-policy</code></td>
<td>Ends the definition of a route policy and exits route policy configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-rpl)# end-policy</code></td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td><code>mpls traffic-eng</code></td>
<td>Enters the MPLS-TE submode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config)# mpls traffic-eng</code></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td><code>attribute-set p2p-te attribute-set-name</code></td>
<td>Specifies the name of the point-to-point traffic-engineered (P2P TE) attribute-set and enters attribute-set configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(configmpls-te)# attribute-set p2p-te low_lat</code></td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td><code>path-selection metric te</code></td>
<td>Sets the path-selection metric to use the TE metric.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-te-attribute-set)# path-selection metric te</code></td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td>Step 10</td>
<td><code>configure</code></td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong></td>
<td>Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.</td>
<td></td>
</tr>
<tr>
<td>router bgp as-number</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong></td>
<td>Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.</td>
<td></td>
</tr>
<tr>
<td>neighbor ip-address</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 13</strong></td>
<td>Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.</td>
<td></td>
</tr>
<tr>
<td>address-family {ipv4</td>
<td>ipv6} unicast</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 14</strong></td>
<td>Applies the specified policy to inbound IPv4 unicast routes.</td>
<td></td>
</tr>
<tr>
<td>route-policy route-policy-name {in</td>
<td>out}</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy BGP_TE in</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>commit</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Configure Explicit BGP SR-TE

Perform this task to configure explicit BGP SR-TE:

**Before You Begin**

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

```bash
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. extcommunity-set opaque **name**
3. **name**
4. end-set
5. route-policy **route-policy-name**
6. end-policy
7. router bgp **as-number**
8. bgp router-id **ip-address**
9. address-family {ipv4 | ipv6} sr-policy
10. exit
11. neighbor **ip-address**
12. remote-as **as-number**
13. address-family {ipv4 | ipv6} unicast
14. route-policy **route-policy-name** {in | out}
15. send-extended-community-ebgp

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>extcommunity-set opaque <strong>name</strong></td>
<td>Defines the color extended community-set.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# extcommunity-set opaque color1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>name</strong></td>
<td>Defines the color extended community-set.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-ext)# 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>end-set</td>
<td>Ends the definition of the extended community-set.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config-ext)# end-set</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>route-policy <strong>route-policy-name</strong></td>
<td>Creates a route policy and enters route policy configuration mode, where you can define the route policy to mark the prefixes with the color extended community value.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP/0/RSP0/CPU0:router(config)# route-policy color</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rpl)# if destination in (5.5.5.1/32) then</td>
<td>Ends the definition of a route policy and exits route policy configuration mode.</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity color color1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rpl-if)# endif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rpl)# end-policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> end-policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-rpl)# end-policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> router bgp as-number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# router bgp 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong> bgp router-id ip-address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.10.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> address-family {ipv4</td>
<td>ipv6} sr-policy</td>
<td>Specifyes either the IPv4 or IPv6 address family and enters address family configuration submode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 sr-policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> exit</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 11</strong> neighbor ip-address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.10.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 12</strong> remote-as as-number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Configure SR-TE Policies

Configure Explicit BGP SR-TE
### Purpose

**Command or Action**

**Step 13**

`address-family {ipv4 | ipv6} unicast`

**Example:**

```
RP/0/RSP0/CPU0:router# config-bgp-nbr
address-family ipv4 unicast
```

**Purpose**

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

**Step 14**

`route-policy route-policy-name {in | out}`

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
route-policy color out
```

**Purpose**

Applies the specified policy to IPv4 unicast routes.

**Step 15**

`send-extended-community-ebgp`

**Example:**

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
send-extended-community-ebgp
```

**Purpose**

Sends extended community attributes to external Border Gateway Protocol (eBGP) neighbors.

---

## 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
```

**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.
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
```

```
Name: tunnel-te1 Destination: 10.1.1.10 Ifthandle: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
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 sbfd</td>
<td>Enters SBFD configuration mode.</td>
</tr>
</tbody>
</table>

Example:
Router_5(config)# sbfd

Step 3 local-discriminator \{ipv4-address | 32-bit-value | dynamic | interface interface\}

Example:
Router_5(config-sbfd)# local-discriminator 1.1.1.5
Router_5(config-sbfd)# local-discriminator 987654321
Router_5(config-sbfd)# local-discriminator dynamic
Router_5(config-sbfd)# local-discriminator interface Loopback0

Step 4 commit

Verify the local discriminator configuration.

Router_5# show bfd target-identifier local

Local Target Identifier Table
-----------------------------
Discr Discr Src VRF Status Flags
----- --------- ------- -------- --------
16843013 Local default enable ---- ia-
987654321 Local default enable ---- y-
2147483649 Local default enable -------d
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Step 2 bfd</td>
<td>Enters BFD configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router_1(config)# bfd</td>
<td></td>
</tr>
<tr>
<td>Step 3 multipath include location node-id</td>
<td>Configures BFD multiple path on specific line card. Any of the configured line cards can be instructed to host a BFD session.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>Router_1(config-bfd)# multipath include location 0/1/CP00</td>
<td></td>
</tr>
<tr>
<td>Router_1(config-bfd)# multipath include location 0/2/CP00</td>
<td></td>
</tr>
<tr>
<td>Router_1(config-bfd)# multipath include location 0/3/CP00</td>
<td></td>
</tr>
</tbody>
</table>
**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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enters SBFD configuration mode.</td>
</tr>
<tr>
<td>Step 2 sbfd</td>
<td>Configures the remote target.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config)# sbfd</td>
</tr>
<tr>
<td>Step 3 remote-target ipv4 ipv4-address</td>
<td>Maps the destination address (Target ID) to a remote discriminator.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-sbfd)# remote-target ipv4 1.1.1.5</td>
</tr>
<tr>
<td>Step 4 remote-discriminator remote-discriminator</td>
<td>Verify the remote discriminator configuration.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-sbfd-nnnn)# remote-discriminator 16843013</td>
</tr>
</tbody>
</table>

Verify the remote discriminator configuration.

Router_1# show bfd target-identifier remote

Remote Target Identifier Table
-----------------------------------
Discr | Discr Src | VRF | TID Type | Status |
----------------- |---------- |----- |--------- |--------|
Target ID | Name
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>interface tunnel-te tunnel-id</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config)# interface tunnel-te1</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>ipv4 unnumbered type interface-path-id</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if)# ipv4 unnumbered loopback0</td>
</tr>
<tr>
<td>Step 4</td>
<td>Command or Action</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>destination <em>ip-address</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if)# destination 1.1.1.5</td>
</tr>
<tr>
<td>Step 5</td>
<td>bfd</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if)# bfd</td>
</tr>
<tr>
<td>Step 6</td>
<td>fast-detect sbfd</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if-tunte-bfd)# fast-detect sbfd</td>
</tr>
<tr>
<td>Step 7</td>
<td>multiplier <em>multiplier</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if-tunte-bfd)# multiplier 2</td>
</tr>
<tr>
<td>Step 8</td>
<td>minimum-interval <em>milliseconds</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if-tunte-bfd)# minimum-interval 50</td>
</tr>
<tr>
<td>Step 9</td>
<td>exit</td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if-tunte-bfd)# exit</td>
</tr>
<tr>
<td>Step 10</td>
<td>path-option <em>preference-priority explicit name path-name segment-routing</em></td>
</tr>
<tr>
<td>Example:</td>
<td>Router_1(config-if)# path-option 1 explicit name rlr6_exp segment-routing</td>
</tr>
</tbody>
</table>

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.
Configure the SBFD Initiator
Configure IOS XR Traffic Controller (XTC)

IOS XR Traffic Controller (XTC) provides stateful path computation element (PCE) functionality by extending the existing IOS-XR PCE functionality with additional capabilities. The current IOS-XR PCE function is part of MPLS-TE which requires an MPLS package. XTC removes this restriction, making PCE functionality part of the base IOS-XR package. XTC is supported on the MPLS data plane and IPv4 control plane.

To install XTC, you need to install an instance of Cisco IOS XRv 9000 Router. Refer to the Cisco IOS XRv 9000 Router Installation and Configuration Guide for more information.

About IOS XR Traffic Controller (XTC)

The path computation element (PCE) describes a set of procedures by which a path computation client (PCC) can report and delegate control of head-end tunnels sourced from the PCC to a PCE peer. The PCE peer can request the PCC to update and modify parameters of label switched paths (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.

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.

XTC learns topology information by way of IGP (OSPF or IS-IS) through BGP-LS.

XTC is capable of computing paths using the following methods:

- TE metric—XTC uses the TE metric in its path calculations to optimize latency.
- IGP metric—XTC uses the IGP metric in its path calculations to optimize reachability.
• Disjointness—XTC uses the disjoint policy to compute two list of segments that steer traffic from two source nodes to two destination nodes along disjoint paths. 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. XTC 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-groupID, a list of segments is computed, encoding the shortest path from the first source to the first destination. When the second 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. The shortest lists of segments is calculated to steer traffic on the computed paths.

Configure PCE

This task explains how to configure PCE.

Before You Begin

Optionally install and configure an instance of Cisco IOS XRv 9000 Router.

SUMMARY STEPS

1. configure
2. pce
3. address ipv4 address
4. state-sync ipv4 address
5. tcp-buffer 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>pce</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>address ipv4 address</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>state-sync ipv4 address</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>tcp-buffer size</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>password {clear</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>segment-routing {strict-sid-only</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong></td>
<td>timers</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
</tbody>
</table>
Configure the Disjoint Policy (Optional)

This task explains how to configure a disjoint policy on the PCE.

**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

<table>
<thead>
<tr>
<th>Step 1</th>
<th><strong>disjoint-path</strong></th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-pce)# disjoint-path</code></td>
<td>Enters disjoint configuration mode.</td>
</tr>
</tbody>
</table>

| Step 2 | **group-id value type {link | node | srlg | srlg-node} [sub-id value]** | Purpose |
|--------|--------------------------------|---------|
| Example: | `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): |
| | | • **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. |

If a pair of paths that meet the requested disjointness level cannot be found, then the paths will automatically fallback to a lower level: |
| | | • If the requested disjointness level is SRLG or node, then link-disjoint paths will be computed. |
| | | • 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. |

<table>
<thead>
<tr>
<th>Step 3</th>
<th><strong>strict</strong></th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td><code>RP/0/RSP0/CPU0:router(config-pce-disjoint)# strict</code></td>
<td>(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.</td>
</tr>
</tbody>
</table>

| Step 4 | **lsp {1 | 2} pcc ipv4 address lsp-name lsp_name [shortest-path]** | Purpose |
|--------|----------------|---------|
| Example: | `RP/0/RSP0/CPU0:router(config-pce-disjoint)# lsp 1 pcc ipv4 192.168.0.1 lsp-name rtrA_t1 shortest-path 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. |

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 first LSP specified.
Configuration Examples for XTC

The XTC configuration examples in this section use the following multi-domain network diagram.

Figure 5: Multi-Domain Network Diagram

The network is divided into three domains. Each node has been assigned a Prefix-SID from the default SRGB. Nodes in different access domains do not have connectivity to each other. XTC-PCE is the centralized PCE for the entire network.
Before you begin, configure BGP-LS address family redistribution on the core routers. This example is for PE111; similar configuration must be applied to the other core routers.

```plaintext
cruter bgp 1
  bgp router-id 3.3.3.111
  address-family ipv4 unicast
    redistribute ospf 1 route-policy loop
  !
  address-family link-state link-state
    !
  neighbor 4.4.4.4
    remote-as 1
    update-source Loopback0
    address-family link-state link-state
    !
  !
  route-policy loop
    if destination in (3.3.3.0/32 ge 24) then
      drop
    endif
    if destination in (0.0.0.0/0 ge 32) then
      set aigp-metric igp-cost
      pass
    else
      drop
    endif
  end-policy
!
commit
derm
```

**Setup XTC: Example**

Complete these tasks on the PCE to setup and enable XTC.
<table>
<thead>
<tr>
<th>Task Number</th>
<th>Task Description</th>
<th>Sample Configuration</th>
<th>Details</th>
</tr>
</thead>
</table>
| 1           | Configure IGP. This step is required for reachability to the BGP-LS peers which use Loopbacks for peering. | conf t  
!  
router isis 3  
is-type level-2-only  
net 49.0001.0000.0000.0004.00  
address-family ipv4 unicast  
metric-style wide  
!  
interface Loopback0  
address-family ipv4 unicast  
!  
interface GigabitEthernet0/0/0/6  
point-to-point  
address-family ipv4 unicast  
!  
commit  
end | Enabling Segment Routing for IS-IS Protocol  
Enabling Segment Routing for OSPF Protocol | |
| 2           | Configure BGP link-state (BGP-LS). XTC learns topology information through BGP-LS. This step configures BGP to receive the topology information. | conf t  
!  
router bgp 1  
bgp router-id 4.4.4.4  
address-family ipv4 unicast  
table-policy fib  
address-family link-state link-state  
!  
neighbor-group epn  
remote-as 1  
update-source Loopback0  
address-family link-state link-state  
!  
neighbor 3.3.3.111  
use neighbor-group epn  
!  
neighbor 3.3.3.112  
use neighbor-group epn  
!  
neighbor 3.3.3.121  
use neighbor-group epn  
!  
neighbor 3.3.3.122  
use neighbor-group epn  
!  
commit  
end | Configure BGP Link-State |
| 3           | Configure static routing for reachability to the PCC clients. The network information is downloaded to the RIB for actual traffic forwarding. | | |
Centralized Inter-Domain Reachability Optimization: Example

The following example shows how to build a policy from A11 to A21 in which XTC uses the IGP metric in its path calculations. Since these are inter-domain paths without any contiguous IGP between them, there is no path currently from A11 to A21. However, if we add up costs along the links (assuming all links are equal cost), the shortest path would be:

A11 → PE111 → PE121 → A21

Configurations on Node A11

Complete these tasks on A11 to use the IGP metric for path calculations.
Configure IOS XR Traffic Controller (XTC)

Centralized Inter-Domain Reachability Optimization: Example

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Task Description</th>
<th>Sample Configuration</th>
</tr>
</thead>
</table>
| 1           | Configure the PCE session by specifying the XTC server in the mpls traffic-eng section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP. | conf t
mpls traffic-eng
pce
peer source ipv4 1.1.1.11
peer ipv4 4.4.4.4
segment-routing
stateful-client
commit
defend |
| 2           | Configure the tunnel interface and set the metric as igp for the PCE computation. When the policy path option attribute is set to use pce, XR knows to contact PCE to get the path. | conf t
interface tunnel-te100
ipv4 unnumbered Loopback0
destination 2.2.2.21
path-selection
metric igp
path-option 1 dynamic pce segment-routing
commit
defend |
| 3           | Configure static routing to push traffic that is destined to the remote PE Loopback0 interface IP using the newly created SR-TE tunnel interface. | conf t
router static
address-family ipv4 unicast
 2.2.2.21/32 tunnel-te100
commit
defend |

Verify the Configuration on Node A11

Verify the configuration by checking the status of the peering session with XTC, the policy status and the path computed by XTC along with the SID-list and label-stack, and the routing entry for the destination.

```bash
RP/0/0/CP00:A11# show mpls traffic-eng pce peer
            Address   Precedence    State   Learned From
-----------------------------------------------
   4.4.4.4   255          Up   Static config

RP/0/0/CP00:A11# show mpls traffic-eng pce tunnels
Tunnel : tunnel-te100
  Destination : 2.2.2.21
  State : up
  Current path option: 1, path learned from PCE 4.4.4.4
  Admin Weight : 21
  Hop Count : 3

RP/0/0/CP00:A11# show mpls traffic-eng tunnels 100
Name: tunnel-te100  Destination: 2.2.2.21  Ifhandle:0xf0
Signalled-Name: A11_t100
```
Centralized Inter-Domain TE-Metric Optimization: Example

The following example shows how to build a policy from A13 to A23 in which XTC uses the TE metric in its path calculations. Since these are inter-domains paths without any contiguous IGP between them, there is no path currently from A13 to A23. Assuming all links have default TE metric, if we add up the TE metrics along the path, the shortest path would be:

A13 → PE112 → PE122 → A23
However, because the link between PE112 and PE122 has a higher TE metric, the policy is:
A13 → PE112 → PE111 → PE121 → PE122 → A23

Configurations on Node A13
Complete these tasks on A13 to use the TE metric for path calculations.
<table>
<thead>
<tr>
<th>Task Number</th>
<th>Task Description</th>
<th>Sample Configuration</th>
</tr>
</thead>
</table>
| 1           | Configure the PCE session by specifying the XTC server in the `mpls traffic-eng` section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP. | conf t  
  mpls traffic-eng  
  pce  
  peer source ipv4 1.1.1.13  
  peer_ipv4 4.4.4.4  
  segment-routing  
  stateful-client  
  !  
  commit  
  end |
| 2           | Configure the tunnel interface and set the metric as `te` for the PCE computation. When the policy path option attribute is set to use `pce`, XR knows to contact PCE to get the path. | conf t  
  interface tunnel-te100  
  ipv4 unnumbered Loopback0  
  destination 2.2.2.23  
  path-selection  
  metric te  
  !  
  path-option 1 dynamic pce segment-routing  
  !  
  commit  
  end |
| 3           | Configure static routing to push traffic that is destined to the remote PE Loopback0 interface IP using the newly created SR-TE tunnel interface. | conf t  
  router static  
  address-family ipv4 unicast  
  2.2.2.23/32 tunnel-te100  
  !  
  !  
  commit  
  end |

**Verify the Configuration on Node A13**

Verify the configuration on node A13 by checking the status of the peering session with XTC, the policy status and the path computed by XTC along with the SID-list and label-stack, and the routing entry for the destination.

```
RP/0/0/CP00:A13# show mpls traffic-eng pce peer  
Address  Precedence  State  Learned From  
-------------------------  ------------  --------  ----------------------  
 4.4.4.4  255  Up  Static config

RP/0/0/CP00:A13# show mpls traffic-eng pce tunnels  
Tunnel : tunnel-te100  
  Destination : 2.2.2.23  
  state : up  
  Current path option: 1, path learned from PCE 4.4.4.4  
  Admin Weight : 21  
  Hop Count : 3

RP/0/0/CP00:A13# show mpls traffic-eng tunnel 100  
Name: tunnel-te100  Destination: 2.2.2.23  Ifhandle:0xb0
```
Centralized LSP Node Disjointness TE-Metric Optimization: Example

The following example shows how to build policies from A11 to A21 and from A13 to A23 using the node disjoint-path type. Both policies use the same group ID (1) and source (0.0.0.1 ) to indicate that the policies are to be grouped together while performing path computations.

When XTC receives a request for the first policy A11 to A21, it checks the group ID and source. Since no policy exists with the same group ID and source combination, XTC calculates a dynamic path for this policy.
When XTC receives a request for the next policy A13 to A23, it again checks the group ID and source. Since XTC already has an existing policy with the same group ID and source combination, it calculates both policy paths, ensuring that there is no common node in the path. It then returns these paths to the head end routers.

Without the disjoint path feature, the paths of both policies share nodes PE111 and PE121 as the mid-point routers:

- A11 → PE111 → PE121 → A21
- A13 → PE112 → PE111 → PE121 → PE122 → A23

However, using the disjoint-path feature, no nodes are shared in the paths:

- A11 → PE111 → PE121 → A21
- A13 → PE112 → PE122 → A23

**Configurations on Node A11**

Complete these tasks on node A11 to use node disjointness for path calculations.
### Configure IOS XR Traffic Controller (XTC)

#### Sample Configuration

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Task Description</th>
<th>Sample Configuration</th>
</tr>
</thead>
</table>
| 1           | Configure the PCE session by specifying the XTC server in the `mpls traffic-eng` section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP. For the disjoint paths feature, we create an attribute-set with **path-option**. Specify **pce** and configure the disjoint-path type as **node**, group-id value as 1, and source as **0.0.0.1**. The group-id and source are used to identify and group different policies across the network. | conf t  
|             |                                                                                 | mpls traffic-eng  
|             |                                                                                 |   pce  
|             |                                                                                 |     peer source ipv4 1.1.1.11  
|             |                                                                                 |     peer ipv4 4.4.4.4  
|             |                                                                                 |     !  
|             |                                                                                 |     segment-routing  
|             |                                                                                 |     stateful-client  
|             |                                                                                 |     !  
|             |                                                                                 |     auto-tunnel p2p  
|             |                                                                                 |       tunnel-id min 1000 max 2000  
|             |                                                                                 |     attribute-set path-option PO.NODE.DISJ  
|             |                                                                                 |     pce  
|             |                                                                                 |       disjoint-path source 0.0.0.1 type node group-id 1  
|             |                                                                                 |     !  
|             |                                                                                 |     !  
|             |                                                                                 |     commit  
|             |                                                                                 |     end  |
| 2           | Configure the tunnel interface and set the metric as **te** for the PCE computation. Configure the path-option to use the attribute-set **PO.NODE.DISJ** (created in the previous step). This attribute-set also specified **pce**, so the PCE will provide the disjoint path. | conf t  
|             |                                                                                 |   interface tunnel-te200  
|             |                                                                                 |     ipv4 unnumbered Loopback0  
|             |                                                                                 |     destination 2.2.2.21  
|             |                                                                                 |     path-selection  
|             |                                                                                 |       metric te  
|             |                                                                                 |     !  
|             |                                                                                 |     path-option 1 dynamic segment-routing attribute-set PO.NODE.DISJ  
|             |                                                                                 |     !  
|             |                                                                                 |     commit  
|             |                                                                                 |     end  |

**Configurations on Node A13**

Complete these tasks on node A13 to use node disjointness for path calculations.
### Configure the PCE Session

Configure the PCE session by specifying the XTC server in the `mpls traffic-eng` section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP.

For the disjoint paths feature, we create an attribute-set with **path-option**. Specify `pce` and configure the disjoint-path type as `node`, group-id value as 1, and source as `0.0.0.1`. The group-id and source are used to identify and group different policies across the network.

```
conf t
!
mpls traffic-eng
    pce
    peer source ipv4 1.1.1.13
    peer ipv4 4.4.4.4
    segment-routing
    stateful-client
    !
    auto-tunnel p2p
tunnel-id min 1000 max 2000
    attribute-set path-option PO.NODE.DISJ
    pce
    disjoint-path source 0.0.0.1 type node group-id 1
    !
commit
end
```

### Configure the Tunnel Interface

Configure the tunnel interface and set the metric as `te` for the PCE computation.

Configure the path-option to use the attribute-set `PO.NODE.DISJ` (created in the previous step). This attribute-set also specified `pce`, so the PCE will provide the disjoint path.

```
conf t
!
interface tunnel-te200
ipv4 unnumbered Loopback0
destination 2.2.2.23
path-selection
    metric te
    path-option 1 dynamic segment-routing attribute-set PO.NODE.DISJ
pce
    disjoint-path source 0.0.0.1 type node group-id 1
    !
commit
end
```

### Verify the Configuration on Node A11

Verify the configuration on node A11 by checking the status of the peering session with XTC, the policy status and the path computed by XTC along with the SID-list and label-stack, and the routing entry for the destination.

```
RP/0/0/CPU0:A11# show mpls traffic-eng pce peer
Address Precedence State Learned From
----------------- ----------------- ------------
4.4.4.4 255 Up Static config

RP/0/0/CPU0:A11# show mpls traffic-eng pce tunnels
Tunnel : tunnel-te100
    Destination : 2.2.2.21
    State : up
    Current path option: 1, path learned from PCE 4.4.4.4
    Admin Weight : 21
    Hop Count : 3

RP/0/0/CPU0:A11# show mpls traffic-eng tunnels 200
Name: tunnel-te200 Destination: 2.2.2.21 Ifhandle:0xd0
```

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

79
Signalled-Name: A11_t200  
Status:  
  Admin: up  Oper: up  Path: valid  Signalling: connected  

Path option 1, (Segment-Routing) type dynamic pce (Basis for Setup, path weight 30)  
  Path-option attribute: PO%2eNODE%2eDISJ  
  G-PID: 0x0000 (derived from egress interface properties)  
  Bandwidth Requested: 0 kbps CT0  
  Creation Time: Sun Jun 4 19:47:19 2017 (1d05h ago)  
Config Parameters:  
  Bandwidth: 0 kbps (CT0)  Priority: 7 7  Affinity: 0x0/0xffff  
  Metric Type: TE (interface)  
  Path Selection:  
    Tiebreaker: Min-fill (default)  
    Protection: any (default)  
    Hop-limit: disabled  
    Cost-limit: disabled  
    Path-invalidation timeout: 10000 msec (default), Action: Tear (default)  
    AutoRoute: disabled  LockDown: disabled  Tunnel class: not set  
    Forward class: 0 (default)  
    Forwarding-Adjacency: disabled  
    Autoroute Destinations: 0  
    Loadshare: 0 equal loadshares  
    Auto-bw: disabled  
    Path Protection: Not Enabled  
    BFD Fast Detection: Disabled  
    Reoptimization after affinity failure: Enabled  
    SRLG discovery: Disabled  
History:  
  Tunnel has been up for: 1d04h (since Sun Jun 04 20:26:26 UTC 2017)  
  Current LSP:  
    Uptime: 1d04h (since Sun Jun 04 20:46:29 UTC 2017)  
    Reopt. LSP:  
      Last Failure:  
        LSP not signalled, has no S2Ls  
        Date/Time: Tue Jun 06 00:46:29 UTC 2017 [00:31:51 ago]  
      Prior LSP:  
        ID: 2  Path Option: 1  
        Removal Trigger: reoptimization completed  

Segment-Routing Path Info (PCE computed path)  
  Segment0[Link]: 11.111.0.11 - 11.111.0.111, Label: 28107  
  Segment1[Node]: 3.3.3.121, Label: 16121  
  Segment2[Link]: 21.121.0.121 - 21.121.0.21, Label: 24000  
Displayed 1 (of 4) heads, 0 (of 0) midpoints, 0 (of 0) tails  
Displayed 1 up, 0 down, 0 recovering, 0 recovered heads  

Verify the Configuration on Node A13  
Verify the configuration on node 13 by checking the status of the peering session with XTC, the policy status and the path computed by XTC along with the SID-list and label-stack, and the routing entry for the destination.  

RP/0/0/CP00:A13# show mpls traffic-eng pce peer  
Address  Precedence  State  Learned From  
-----------------  ------------  ------------  -------------------  
   4.4.4.4        255        Up  Static config  
RP/0/0/CP00:A13# show mpls traffic-eng pce tunnels  
Tunnel : tunnel-te100  
  Destination : 2.2.2.23  
  State : up  
  Current path option: 1, path learned from PCE 4.4.4.4  
  Admin Weight : 21  
  Hop Count : 3  
RP/0/0/CP00:A13# show mpls traffic-eng tunnels 200  
Name: tunnel-te200  Destination: 2.2.2.23  Ifhandle:0x90
Centralized LSP Link Disjointness TE-Metric Optimization: Example

The following example shows how to build two policies from A12 to A22 using the link disjoint-path type. Both policies use the same group ID (2) and source (0.0.0.2) to indicate that the policies are to be grouped together while performing path computations.

When XTC receives a request for the first policy A12 to A22, it checks the group ID and source. Since no policy exists with the same group ID and source combination, XTC calculates a dynamic path for this policy. When XTC receives a request for the next policy A12 to A23, it again checks the group ID and source. Since XTC already has an existing policy with the same group ID and source combination, it calculates both policy paths, ensuring that there is no common link in the path. It then returns these paths to the head end routers.

Without the disjoint path feature, the dynamic paths of both policies would be the same:
A12 $\rightarrow$ A11 $\rightarrow$ PE111 $\rightarrow$ PE121 $\rightarrow$ A21 $\rightarrow$ A22
However, using the disjoint-path feature, no links are shared in the paths:

A12 → A11 → PE111 → PE121 → A21 → A22
A12 → A13 → PE112 → PE122 → A23 → A22

**Configurations on Node A12**

Complete these tasks on node A12 to use link disjointness for path calculations.
### Configure the PCE session

Configure the PCE session by specifying the XTC server in the `mpls traffic-eng` section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP.

For the disjoint paths feature, we create an attribute-set with `path-option`. Specify `pce` and configure the disjoint-path type as **node**, group-id value as 2, and source as **0.0.0.2**. The group-id and source are used to identify and group different policies across the network.

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Task Description</th>
<th>Sample Configuration</th>
</tr>
</thead>
</table>
| 1           | Configure the PCE session by specifying the XTC server in the `mpls traffic-eng` section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP. For the disjoint paths feature, we create an attribute-set with `path-option`. Specify `pce` and configure the disjoint-path type as **node**, group-id value as 2, and source as **0.0.0.2**. The group-id and source are used to identify and group different policies across the network. | conf t  
! mpls traffic-eng  
pce  
peer source ipv4 1.1.1.12  
peer ipv4 4.4.4.4  
! segment-routing  
stateful-client  
! auto-tunnel p2p  
tunnel-id min 1000 max 2000  
! attribute-set path-option PO.LINK.DISJ  
pce  
disjoint-path source 0.0.0.2 type link group-id 2  
!  
! commit  
end |
| 2           | Configure the tunnel interfaces and set the metric as **te** for the PCE computation. Configure the `path-option` to use the attribute-set **PO.LINK.DISJ** (created in the previous step). This attribute-set also specified `pce`, so the PCE will provide the disjoint path. | conf t  
! interface tunnel-te201  
ipv4 unnumbered Loopback0  
destination 2.2.2.22  
path-selection  
metric te  
! path-option 1 dynamic segment-routing attribute-set PO.LINK.DISJ  
! interface tunnel-te202  
ipv4 unnumbered Loopback0  
destination 2.2.2.22  
path-selection  
metric te  
! path-option 1 dynamic segment-routing attribute-set PO.LINK.DISJ  
! commit  
end |

#### Verify the Configuration on Node A12

Verify the configuration on node A12 by checking the status of the peering session with XTC, the policy status and the path computed by XTC along with the SID-list and label-stack, and the routing entry for the destination.

```
RP/0/0/CPU0:A12# show mpls traffic-eng pce peer
Address  Precedence  State    Learned From
----------  ---------  -------  ---------------------
4.4.4.4     255        Up       Static config

RP/0/0/CPU0:A12# show mpls traffic-eng pce tunnels
Tunnel : tunnel-te201
          Destination : 2.2.2.22
```
State : up
Current path option: 1, path learned from PCE 4.4.4.4
Admin Weight : 50
Hop Count : 5
Tunnel : tunnel-te202
Destination : 2.2.2.22
State : up
Current path option: 1, path learned from PCE 4.4.4.4
Admin Weight : 50040
Hop Count : 5
RP/0/0/CPU0:A12# show mpls traffic-eng tunnels 201
Name: tunnel-te201 Destination: 2.2.2.22 Ifhandle:0x90
Signalled-Name: A12_t201
Status:
  Admin: up Oper: up Path: valid Signalling: connected
  path option 1, (Segment-Routing) type dynamic pce (Basis for Setup, path weight 50)
  Path-option attribute: PO%2eLINK%2eDISJ
  G-PID: 0x0800 (derived from egress interface properties)
  Creation Time: Sun Jun 4 20:47:49 2017 (1d04h ago)
  Config Parameters:
    Bandwidth: 0 kbps (CT0) Priority: 7 7 Affinity: 0x0/0xffffffff
  Metric Type: TE (interface)
  Path Selection:
    Tiebreaker: Min-fill (default)
    Protection: any (default)
    Hop-limit: disabled
    Cost-limit: disabled
    Path-invalidation timeout: 10000 ms (default), Action: Tear (default)
    AutoRoute: disabled LockDown: disabled Tunnel class: not set
    Forward class: 0 (default)
    Forwarding-Adjacency: disabled
    Autoroute Destinations: 0
    Loadshare: 0 equal loadshares
    Auto-bw: disabled
    Path Protection: Not Enabled
    BFD Fast Detection: Disabled
    Reoptimization after affinity failure: Enabled
    SRLG discovery: Disabled
  History:
    Tunnel has been up for: 1d04h (since Sun Jun 04 20:49:50 UTC 2017)
    Current LSP:
      Uptime: 00:56:03 (since Tue Jun 06 00:46:39 UTC 2017)
    Reopt. LSP:
      Last Failure:
        LSP not signalled, has no S2Ls
      Date/Time: Tue Jun 06 00:46:36 UTC 2017 [00:56:06 ago]
      Prior LSP:
        ID: 59 Path Option: 1
        Removal Trigger: reoptimization completed
  Segment-Routing Path Info (PCE computed path)
    Segment0[Link]: 11.12.1.12 - 11.12.1.11, Label: 28097
    Segment1[Link]: 11.111.0.111 - 11.111.0.111, Label: 28097
    Segment2[Node]: 3.3.3.121, Label: 16121
    Segment3[Link]: 21.121.0.121 - 21.121.0.21, Label: 24000
    Segment4[Link]: 21.22.1.21 - 21.22.1.22, Label: 28097
  Displayed 1 (of 3) heads, 0 (of 0) midpoints, 0 (of 0) tails
  Displayed 1 up, 0 down, 0 recovering, 0 recovered heads
RP/0/0/CPU0:A12# show mpls traffic-eng tunnels 202
Name: tunnel-te202 Destination: 2.2.2.22 Ifhandle:0xb0
Signalled-Name: A12_t202
Status:
  Admin: up Oper: up Path: valid Signalling: connected
  path option 1, (Segment-Routing) type dynamic pce (Basis for Setup, path weight 50040)
Centralized LSP SRLG Disjointness TE-Metric Optimization: Example

The following example shows how to build two policies from A11 to A23 using the shared link risk group (SRLG) disjoint-path type. Both policies use the same group ID (3) and source (0.0.0.3) to indicate that the policies are to be grouped together while performing path computations.

**Note**
SRLG disjointness can be considered as SRLG + LINK disjoint, and both conditions must be met to have feasible paths with no common links.

There are four links in the same SRLG: Gig0/0/0/2 and Gig0/0/0/3 between PE111 and PE112, and between PE121 and PE122. The value of these links is set to 100.

When XTC receives a request for the first policy A11 to A23, it checks the group ID and source. Since no policy exists with the same group ID and source combination, XTC calculates a dynamic path for this policy.

When XTC receives a request for the next policy A11 to A23, it again checks the group ID and source. Since XTC already has an existing policy with the same group ID and source combination, it calculates both policy
paths, ensuring that there is no link in the same SRLG in the path. It then returns these paths to the head end routers.

If the disjoint path feature is not configured, the dynamic paths of both policies are the same:

A11 → PE111 → PE121 → PE122 → A23

However, when the disjoint-path feature is configured, no two links use the same SRLG in the paths:

A11 → PE111 → PE121 → A21 → A22 → A23
A11 → A12 → A13 → PE112 → PE122 → A23

Configuration on PE111, PE112, PE121, PE122

Configure SRLG on the nodes in the core (PE111, PE112, PE121, PE122)
Configure SRLG on the nodes in the core so XTC knows which links are in the same group.

```
conf t
! srlg
interface GigabitEthernet0/0/0/2
  8 value 100
  name CORE_TOP_DOWN
!
interface GigabitEthernet0/0/0/3
  8 value 100
  name CORE_TOP_DOWN
!
commit
end
```

**Configurations on Node A11**

Complete these tasks on node A11 to use SRLG disjointness for path calculations.
### Task Number | Task Description | Sample Configuration
---|---|---
1 | Configure the PCE session by specifying the XTC server in the `mpls traffic-eng` section. The source IP is typically the Loopback0 interface IP. This is a stateful connection and will stay connected to the XTC node by way of TCP. For the disjoint paths feature, we create an attribute-set with `path-option`. Specify `pce` and configure the disjoint-path type as `srlg`, group-id value as `3`, and source as `0.0.0.3`. The group-id and source are used to identify and group different policies across the network. | conf t  
!  
`mpls traffic-eng`  
`pce`  
`peer source ipv4 1.1.1.11`  
`peer ipv4 4.4.4.4`  
!  
`segment-routing`  
`stateful-client`  
!  
`auto-tunnel p2p`  
`tunnel-id min 1000 max 2000`  
!  
`attribute-set path-option PO.SRLG.DISJ`  
`pce`  
`disjoint-path source 0.0.0.3 type srlg group-id 3`  
!  
!  
commit  
end 

2 | Configure the tunnel interfaces and set the metric as `te` for the PCE computation. Configure the path-option to use the attribute-set `PO.SRLG.DISJ` (created in the previous step). This attribute-set also specified `pce`, so the PCE will provide the disjoint path. | conf t  
!  
`interface tunnel-te201`  
`ipv4 unnumbered Loopback0`  
`destination 2.2.2.23`  
`path-selection`  
`metric te`  
`!  
`path-option 1 dynamic segment-routing attribute-set PO.SRLG.DISJ`  
`!  
`interface tunnel-te202`  
`ipv4 unnumbered Loopback0`  
`destination 2.2.2.23`  
`path-selection`  
`metric te`  
`!  
`path-option 1 dynamic segment-routing attribute-set PO.SRLG.DISJ`  
!commit  
end 

### Verify the Configuration on Node A11
Verify the configuration on node A11 by checking the status of the peering session with XTC, the policy status and the path computed by XTC along with the SID-list and label-stack, and the routing entry for the destination.

```
RP/0/0/CPU0:A11# show mpls traffic-eng pce peer  
Address  Precedence  State  Learned From  
4.4.4.4  255  Up  Static config

RP/0/0/CPU0:A11# show mpls traffic-eng pce tunnels  
Tunnel : tunnel-te201  
Destination : 2.2.2.23
```
State : up
Current path option: 1, path learned from PCE 4.4.4.4
Admin Weight : 50
Hop Count : 4

Tunnel : tunnel-te202
Destination : 2.2.2.23
State : up
Current path option: 1, path learned from PCE 4.4.4.4
Admin Weight : 50040
Hop Count : 5

RP/0/0/CPU0:A11# show mpls traffic-eng tunnels 201

Name: tunnel-te201 Destination: 2.2.2.23 Ifhandle:0x170
Signalled-Name: A11_t201
Status:
Admin: up Oper: up Path: valid Signalling: connected

path option 1, (Segment-Routing) type dynamic pce (Basis for Setup, path weight 50)
Path-option attribute: PO%2eSRLG%2eDISJ
G-PID: 0x0800 (derived from egress interface properties)
Bandwidth Requested: 0 kbps CT0
Creation Time: Tue Jun  6 18:59:07 2017 (1w1d ago)
Config Parameters:
Bandwidth: 0 kbps (CT0) Priority: 7 7 Affinity: 0x0/0xffff
Metric Type: TE (interface)
Path Selection:
Tiebreaker: Min-fill (default)
Protection: any (default)
Hop-limit: disabled
Cost-limit: disabled
Path-invalidation timeout: 10000 msec (default), Action: Tear (default)
AutoRoute: disabled LockDown: disabled Tunnel class: not set
Forward class: 0 (default)
Forwarding-Adjacency: disabled
Autoroute Destinations: 0
Loadshare: 0 equal loadshares
Auto-bw: disabled
Path Protection: Not Enabled
BFD Fast Detection: Disabled
Reoptimization after affinity failure: Enabled
SRLG discovery: Disabled
History:
Tunnel has been up for: 6d21h (since Wed Jun 07 22:25:32 UTC 2017)
Current LSP:
Uptime: 00:39:00 (since Wed Jun 14 18:46:36 UTC 2017)
Reopt. LSP:
Last Failure:
LSF not signalled, has no S2Ls
Date/Time: Wed Jun 14 18:46:32 UTC 2017 [00:39:04 ago]
Prior LSP:
ID: 149 Path Option: 1
Removal Trigger: reoptimization completed

Segment-Routing Path Info (PCE computed path)
Segment0[Link]: 11.111.0.11 - 11.111.0.111, Label: 28107
Segment1[Node]: 3.3.3.121, Label: 16121
Segment2[Link]: 21.121.0.121 - 21.121.0.21, Label: 24000
Segment3[Node]: 2.2.2.23, Label: 16023
Displayed 1 (of 4) heads, 0 (of 0) midpoints, 0 (of 0) tails
Displayed 1 up, 0 down, 0 recovering, 0 recovered heads

RP/0/0/CPU0:A11# show mpls traffic-eng tunnels 202

Name: tunnel-te202 Destination: 2.2.2.23 Ifhandle:0x110
Signalled-Name: A11_t202
Status:
Admin: up Oper: up Path: valid Signalling: connected

Segment Routing Configuration Guide for Cisco ASR 9000 Series Routers, IOS XR Release 6.2.x
path option 1, (Segment-Routing) type dynamic pce (Basis for Setup, path weight 50040)
  Path-option attribute: PO%eSRLG%eDISJ
  G-PID: 0x0800 (derived from egress interface properties)
  Bandwidth Requested: 0 kbps CT0
  Creation Time: Sun Jun 4 19:47:19 2017 (1w2d ago)
Config Parameters:
  Bandwidth: 0 kbps (CT0) Priority: 7 7 Affinity: 0x0/0xffff
Metric Type: TE (interface)
  Path Selection:
    Tiebreaker: Min-fill (default)
    Protection: any (default)
    Hop-limit: disabled
    Cost-limit: disabled
    Path-invalidation timeout: 10000 msec (default), Action: Tear (default)
    AutoRoute: disabled LockDown: disabled Tunnel class: not set
    Forward class: 0 (default)
    Forwarding-Adjacency: disabled
    Autoroute Destinations: 0
    Loadshare: 0 equal loadshares
    Auto-bw: disabled
    Path Protection: Not Enabled
    BFD Fast Detection: Disabled
    Reoptimization after affinity failure: Enabled
    SRLG discovery: Disabled
History:
  Tunnel has been up for: 6d21h (since Wed Jun 07 22:25:32 UTC 2017)
  Current LSP:
    Uptime: 1d20h (since Mon Jun 12 22:41:56 UTC 2017)
    Reopt. LSP:
    Last Failure:
      LSP not signalled, has no S2Ls
      Date/Time: Wed Jun 14 18:46:32 UTC 2017 [00:40:05 ago]
  Prior LSP:
    ID: 59 Path Option: 1
    Removal Trigger: reoptimization completed
Segment-Routing Path Info (PCE computed path)
  Segment0[Link]: 11.12.1.11 - 11.12.1.12, Label: 28115
  Segment1[Link]: 12.13.2.12 - 12.13.2.13, Label: 28101
  Segment2[Link]: 13.112.0.13 - 13.112.0.112, Label: 28107
  Segment3[Node]: 3.3.3.122, Label: 16122
  Segment4[Link]: 23.122.0.122 - 23.122.0.23, Label: 28096
  Displayed 1 (of 4) heads, 0 (of 0) midpoints, 0 (of 0) tails
  Displayed 1 up, 0 down, 0 recovering, 0 recovered heads
RP/0/0/CPU0:A11#
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. The goal of TI-LFA is to reduce the packet loss that results while routers converge after a topology change due to a link 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.

TI-LFA provides link protection. The link is excluded during the post convergence backup path calculation.

TI-LFA node protection provides protection from node failures. The neighbor node is excluded during the post convergence backup path calculation.

Shared Risk Link Groups (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.

For IS-IS, TI-LFA node protection and SRLG protection can be configured on the interface or the instance. For OSPF, TI-LFA node protection and SRLG protection are configured on the interface.

- Configuring TI-LFA for IS-IS, page 91
- Configuring TI-LFA for OSPF, page 93
- TI-LFA Node and SRLG Protection: Examples, page 95
- Configuring and Verifying TI-LFA: Example, page 96

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 LSPs are configured.

SUMMARY STEPS

1. `configure`
2. `router isis instance-id`
3. `interface type interface-path-id`
4. `address-family { ipv4 | ipv6 } [ 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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure</code></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>router isis instance-id</code> &lt;br&gt; <strong>Example:</strong>&lt;br&gt; RP/0/RSP0/CPU0:router(config)# router isis 1</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>interface type interface-path-id</code> &lt;br&gt; <strong>Example:</strong>&lt;br&gt; RP/0/RSP0/CPU0:router(config-isis)# interface GigabitEthernet0/0/2/1&lt;br&gt; RP/0/RSP0/CPU0:router(config-isis)# interface Bundle-Ether1</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>`address-family { ipv4</td>
</tr>
</tbody>
</table>
### Configure Topology-Independent Loop-Free Alternate (TI-LFA)

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 OSPF.
- Segment routing LSPs are configured.

---

#### Configuring TI-LFA for OSPF

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 5</strong></td>
<td>Enables per-prefix fast reroute.</td>
</tr>
<tr>
<td>fast-reroute per-prefix</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>Enables per-prefix TI-LFA fast reroute link protection.</td>
</tr>
<tr>
<td>fast-reroute per-prefix ti-lfa</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix ti-lfa</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>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. The same attribute cannot be configured more than once on an interface.</td>
</tr>
<tr>
<td>fast-reroute per-prefix tiebreaker {node-protecting</td>
<td>srlg-disjoint} index priority</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config-isis-if-af)# fast-reroute per-prefix srlg-disjoint index 100</td>
<td>Note</td>
</tr>
</tbody>
</table>

TI-LFA has been successfully configured for segment routing.
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>router ospf <em>process-name</em></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>area <em>area-id</em></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf)# area 1</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>interface <em>type interface-path-id</em></td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar)# interface GigabitEthernet0/0/2/1</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>fast-reroute per-prefix</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix</td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>fast-reroute per-prefix ti-lfa</td>
</tr>
<tr>
<td>Example:</td>
<td>RP/0/RSP0/CPU0:router(config-ospf-ar-if)# fast-reroute per-prefix ti-lfa</td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>fast-reroute per-prefix tiebreaker {node-protecting</td>
</tr>
</tbody>
</table>
Purpose

Example:

```
RP/0/RSP0/CPU0:router(config-isis-ar-if)>
fasteroute per-prefix srlg-disjoint index 100
```

Link protection always has a lower priority than node or SRLG protection.

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
      fasteroute per-prefix
      fasteroute per-prefix ti-lfa
      fasteroute 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
      fasteroute per-prefix
      fasteroute per-prefix ti-lfa
      fasteroute 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
      fasteroute per-prefix
      fasteroute per-prefix ti-lfa
      fasteroute per-prefix tiebreaker node-protecting index 100
      fasteroute per-prefix tiebreaker srlg-disjoint index 200
```

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.
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.

Configuring and Verifying TI-LFA: Example

In this example, we configure TI-LFA for segment routing TE tunnels using single or disjoint PQ nodes. The following figures show the two topologies used in this example:

- Topology 1 uses a single PQ Node, and therefore has two SIDs; from the source router, R1, through the PQ Node, to the destination router, R5.

Figure 6: Topology 1: Single PQ Node
• Topology 2 uses disjoint PQ Nodes, and therefore consists of three SIDs; from the source router, R1, through the P Node and the Q Node to the destination router, R5.

**Figure 7: Topology 2: Disjoint PQ Nodes**

---

**Step 1**  Configure TI-LFA for IS-IS or OSPF on the source router (R1) interface connecting to the destination router (R5):

- **For IS-IS**
  ```
  RP/0/RSP0/CPU0:R1(config)# router isis 1
  RP/0/RSP0/CPU0:R1(config-isis)# interface TenGigE0/0/0/2/1
  RP/0/RSP0/CPU0:R1(config-isis-if)# point-to-point
  RP/0/RSP0/CPU0:R1(config-isis-if)# address-family ipv4 unicast
  RP/0/RSP0/CPU0:R1(config-isis-if)# fast-reroute per-prefix
  RP/0/RSP0/CPU0:R1(config-isis-if)# fast-reroute per-prefix ti-lfa
  RP/0/RSP0/CPU0:R1(config-isis-if)# exit
  ```

- **For OSPF**
  ```
  RP/0/RSP0/CPU0:R1(config)# router ospf 1
  RP/0/RSP0/CPU0:R1(config-ospf)# area 0
  RP/0/RSP0/CPU0:R1(config-ospf-ar)# interface TenGigE0/0/0/2/1
  RP/0/RSP0/CPU0:R1(config-ospf-ar-if)# fast-reroute per-prefix
  RP/0/RSP0/CPU0:R1(config-ospf-ar-if)# fast-reroute per-prefix ti-lfa
  RP/0/RSP0/CPU0:R1(config-ospf-ar-if)# exit
  ```

**Note**  For this example, we configured TI-LFA on the specific interface. TI-LFA can be configured on the instance or area; all interfaces in the instance or area would inherit the configuration.

**Step 2**  Configure autoroute announce on the segment routing tunnel interface connecting R1 to R5.
Example:

RP/0/RSP0/CPU0:R1(config)# interface tunnel-te1
RP/0/RSP0/CPU0:R1(config-if)# ipv4 unnumbered Loopback0
RP/0/RSP0/CPU0:R1(config-if)# autoroute announce
RP/0/RSP0/CPU0:R1(config-if)# destination 192.168.5.1
RP/0/RSP0/CPU0:R1(config-if)# path-option 1 dynamic segment-routing

Dynamic segment routing path option is configured to use adjacency SIDs for segment routing.

**Step 3**

Define the Segment Routing Global Block (SRGB) for the network.

Example:

RP/0/RSP0/CPU0:R1(config-isis)# segment-routing global-block 50000 60000

If we configure segment routing to be used with prefix SIDs, the segment routing labels are assigned from the defined global block. In this example, we configure the source router to use adjacency SIDs dynamically, and therefore, the label assignment does not use the SRGB.

**Step 4**

Commit your configuration.

Example:

RP/0/RSP0/CPU0:R1# commit

**Step 5**

Verify the IP FRR protection on the path between the source and destination router. The following output is for a single PQ node topology.

Example:

RP/0/RSP0/CPU0:R1# show mpls traffic-eng forwarding tunnels 1 detail
Tunnel Name Label Outgoing Interface Outgoing Next Hop Switched Bytes
------------- ----------- ------------ --------------- ------------
    te1 (SR) Pop Te0/1/1/1.100 10.15.1.2 31340256

Updated: Aug 28 10:21:27.763
Path Flags: 0x400 [ BKUP-IDX:1 (0x0) ]
Label Stack (Top -> Bottom): { Imp-Null }
NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 1, Weight: 0
MAC/Encaps: 18/18, MTU: 1496
Packets Switched: 26616

50103 Te0/3/0/11.100 10.12.2.2 10.15.1.2 31340256

Updated: Aug 28 10:21:27.763
Path Flags: 0x100 [ BKUP, NoFwd ]
Label Stack (Top -> Bottom): { 50103, 50105 }
NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 0, Weight: 0
MAC/Encaps: 18/26, MTU: 1496
Packets Switched: 0
(!): FRR pure backup

Interface Handle: 0x080000120, Local Label: 24002
Forwarding Class: 0, Weight: 0
Packets/Bytes Switched: 34727459/40968290594

The following output is for disjoint PQ nodes.
Example:

RP/0/RSP0/CPU0:R1# show mpls forwarding tunnels 1 detail

<table>
<thead>
<tr>
<th>Tunnel Name</th>
<th>Outgoing Label</th>
<th>Outgoing Interface</th>
<th>Next Hop</th>
<th>Bytes Switched</th>
</tr>
</thead>
<tbody>
<tr>
<td>tt1</td>
<td>Pop 10.15.1.2</td>
<td>Te0/1/1/1.100</td>
<td>65361590</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Updated: Aug 31 07:52:17.630
Path Flags: 0x400 [ BKUP-IDX:1 (0x0) ]
Version: 42799904, Priority: 2
Label Stack (Top -> Bottom): { Imp-Null }
NHID: 0x0, Encap-ID: N/A, Path idx: 0, Backup path idx: 1, Weight: 0
MAC/Encaps: 18/18, MTU: 1496
Packets Switched: 55581

50103
<table>
<thead>
<tr>
<th>Te0/0/0/2/1 10.12.100.2</th>
<th>0 (!)</th>
</tr>
</thead>
</table>

Updated: Aug 31 07:52:17.630
Path Flags: 0x100 [ BKUP, NoFwd ]
Version: 42799904, Priority: 2
Label Stack (Top -> Bottom): { 50103 50104 50105 }
NHID: 0x0, Encap-ID: N/A, Path idx: 1, Backup path idx: 0, Weight: 0
MAC/Encaps: 14/26, MTU: 1500
Packets Switched: 0
(!): FRR pure backup

Interface Handle: 0x08000120, Local Label: 24029
Forwarding Class: 0, Weight: 0
Packets/Bytes Switched: 97227973/114534209178

The output for a single PQ node confirms that the primary path from R1 to R5 (label is popped as there are no transit routers) is protected by the backup path with the labels 50103 (for the path from router R1 to the PQ Node, router R3) and 50105 (for the path from router R3 to the destination router R5).

The output for disjoint PQ nodes confirms that the primary path from R1 to R5 (label is popped as there are no transit routers) is protected by the backup path with the labels 50103 (for the path from router R1 to the PQ Node, router R3), 50104 (for the path from PQ Node (router R3) to Q Node, router R4) and 50105 (for the path from router R4 to the destination router R5).

Note When dynamic segment routing is configured on a router, adjacency SIDs are used and the labels are not picked from the SRGB (as shown, in this example, by the local label of 24002).

TI-LFA has been successfully configured for segment routing.
Configure Topology-Independent Loop-Free Alternate (TI-LFA)

Configuring and Verifying TI-LFA: Example
CHAPTER 10

Configure Segment Routing Microloop Avoidance

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

- About Segment Routing Microloop Avoidance, page 101
- Configure Segment Routing Microloop Avoidance for IS-IS, page 101
- Configure Segment Routing Microloop Avoidance for OSPF, page 102

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.

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.
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2 router isis instance-id</td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode. You can change the level of routing to be performed by a particular routing instance by using the is-type router configuration command.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router isis 1</td>
<td></td>
</tr>
<tr>
<td>Step 3 address-family ipv4 [ unicast ]</td>
<td>Specifies the IPv4 address family and enters router address family configuration mode.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis-af)# address-family ipv4 unicast</td>
<td></td>
</tr>
<tr>
<td>Step 4 microloop avoidance segment-routing</td>
<td>Enables Segment Routing Microloop Avoidance.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis-af)# microloop avoidance segment-routing</td>
<td></td>
</tr>
<tr>
<td>Step 5 microloop avoidance rib-update-delay delay-time</td>
<td>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.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-isis-af)# microloop avoidance rib-update-delay 3000</td>
<td></td>
</tr>
</tbody>
</table>

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.

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure</td>
<td>Enables OSPF routing for the specified routing process, and places the router in router configuration mode.</td>
</tr>
<tr>
<td>Step 2 router ospf process-name</td>
<td>Enables Segment Routing Microloop Avoidance.</td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config)# router ospf 1</td>
<td>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.</td>
</tr>
<tr>
<td>Step 3 microloop avoidance segment-routing</td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-ospf)# microloop avoidance segment-routing</td>
<td></td>
</tr>
<tr>
<td>Step 4 microloop avoidance rib-update-delay delay-time</td>
<td></td>
</tr>
<tr>
<td>Example: RP/0/RSP0/CPU0:router(config-ospf)# microloop avoidance rib-update-delay 3000</td>
<td></td>
</tr>
</tbody>
</table>
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, page 105
- Segment Routing and LDP Interoperability, page 106
- Configuring Mapping Server, page 108
- Enable Mapping Advertisement, page 110
- Enable Mapping Client, page 112

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 1037.) P6 swaps 103 for 1037 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. commit
8. show segment-routing mapping-server prefix-sid-map [ipv4 | ipv6] [detail]

#### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>segment-routing</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config)# segment-routing</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>mapping-server</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-sr)# mapping-server</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>prefix-sid-map</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RSP0/CPU0:router(config-sr-ms)# prefix-sid-map</td>
</tr>
</tbody>
</table>

**Note**: Two-way prefix SID can be enabled directly under IS-IS or through a mapping server.
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><code>address-family ipv4</code></td>
<td>Specifies the address-family for IS-IS.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This example shows the address-family for ipv4:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-sr-ms-map)# address-family ipv4</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This example shows the address-family for ipv6:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-sr-ms-map)# address-family ipv6</code></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><code>ip-address/prefix-length first-SID-value range range</code></td>
<td>Adds SID-mapping entries in the active local mapping policy. In the configured example:</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-sr-ms-map-af)# 10.1.1.1/32 10 range 200</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router(config-sr-ms-map-af)# 20.1.0.0/16 400 range 300</code></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><code>commit</code></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>`show segment-routing mapping-server prefix-sid-map [ipv4</td>
<td>ipv6] [detail]`</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router# show segment-routing mapping-server prefix-sid-map ipv4</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Prefix   SID Index  Range  Flags</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>20.1.1.0/24  400  300</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>10.1.1.1/32  10  200</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of mapping entries: 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RP/0/RSP0/CPU0:router# show segment-routing mapping-server prefix-sid-map ipv4 detail</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Prefix</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>20.1.1.0/24</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>SID Index:  400</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Range:  300</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Last Prefix:  20.2.44.0/24</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Last SID Index: 699</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Flags:</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>10.1.1.1/32</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>SID Index:  10</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Range:  200</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Last Prefix:  10.1.1.200/32</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Last SID Index: 209</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>Flags:</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of mapping entries: 2</td>
<td></td>
</tr>
</tbody>
</table>
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. `commit`
5. `show isis database verbose`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><code>router isis instance-id</code></td>
<td>Enables IS-IS routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>You can change the level of routing to be performed by a particular routing instance by using the <strong>is-type</strong> router configuration command.</td>
</tr>
<tr>
<td>RP/0/RSP0/CPU0:router(config)# <code>router isis 1</code></td>
<td></td>
</tr>
</tbody>
</table>

| **Step 2** | Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. |
| `address-family { ipv4 | ipv6 } [ unicast ]` | Specifies the IPv4 or IPv6 address family, and enters router address family configuration mode. |
| **Example:** | The following is an example for ipv4 address family: |
| RP/0/RSP0/CPU0:router(config-isis)# `address-family` | |
| `ipv4 unicast` | |
Configure Segment Routing Mapping Server

Configure Mapping Advertisement for OSPF

SUMMARY STEPS

1. `router ospf process-name`
2. `segment-routing prefix-sid-map advertise-local`
3. `commit`
4. `show ospf database opaque-area`

DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td><code>router ospf process-name</code></td>
<td>Enables OSPF routing for the specified routing instance, and places the router in router configuration mode.</td>
</tr>
</tbody>
</table>

Example:

```
RP/0/RSP0/CPU0:router(config) # router ospf 1
```
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

Collecting Traffic Statistics

The traffic collector is designed to help users understand traffic patterns on a router. Traffic collector collects packet and byte statistics from router components such as prefix counters, tunnel counters, and the traffic matrix counter, which 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.

- Traffic Collector Process, page 113
- Configuring Traffic Collector, page 114
- Displaying Traffic Information, page 115

Traffic Collector Process

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
# 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 l3 interface name`
7. `commit`

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>configure</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>traffic-collector</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config) # traffic-collector</td>
</tr>
<tr>
<td>Purpose</td>
<td>Enables traffic collector and places the router in traffic collector configuration mode.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>statistics collection-interval value</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-tc) # statistics collection-interval 5</td>
</tr>
<tr>
<td>Purpose</td>
<td>(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.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>statistics history-size value</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-tc) # statistics history-size 10</td>
</tr>
<tr>
<td>Purpose</td>
<td>(Optional) Specifies the number of entries kept in the history database. Valid values are from 1 to 10. The default is 5.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>statistics history-timeout value</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>RP/0/RP0/CPU0:router(config-tc) # statistics history-timeout 24</td>
</tr>
<tr>
<td>Purpose</td>
<td>(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.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>Enter 0 to disable the history timeout. (No history is retained.)</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Step 6 <strong>interface l3 interface name</strong></td>
<td>Identifies interfaces that handle external traffic. Only l3 interfaces are supported for external traffic.</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:router(config-tc)# interface TenGigE0/1/0/3</td>
<td></td>
</tr>
<tr>
<td>Step 7 <strong>commit</strong></td>
<td></td>
</tr>
</tbody>
</table>

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/RP0/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/RP0/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
  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: 95288464810
  22:59 - 23:00: Packets 9579889, Bytes: 9407450998
  ```
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 tunnel:

```
RP/0/RSP0/CPU0:router# show traffic-collector counters tunnels tunnel-te 1 detail
Tunnel: tt1 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
  23:11 – 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 tunnel.
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 SID and Nil-FEC (forwarding equivalence classes) LSP Ping and Traceroute functionality.

- MPLS Ping and Traceroute for BGP and IGP Prefix-SID, page 117
- Examples: MPLS Ping, Traceroute, and Tree Trace for Prefix-SID, page 118
- MPLS LSP Ping and Traceroute Nil FEC Target, page 119
- Examples: LSP Ping and Traceroute for Nil_FEC Target, page 120

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:

```
1.1.1.1
 R1
     /   \\    /  \\
    R2   R3  R4
            |
           1.1.1.4

R5  R6
```

**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 tlv, 'N' - no rx label,
'P' - no rx intf label prot, 'p' - premature termination of LSP,
'R' - transit router, 'T' - 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 tlv, 'N' - no rx label,
'P' - no rx intf label prot, 'p' - premature termination of LSP,
'R' - transit router, 'T' - unknown upstream index,
```

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

<table>
<thead>
<tr>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ping mpls nil-fec labels {label[,...,label]} [output {interface tx-interface} [nexthop nexthop-ip-addr]]</td>
</tr>
<tr>
<td>traceroute mpls nil-fec labels {label[,...,label]} [output {interface tx-interface} [nexthop nexthop-ip-addr]]</td>
</tr>
</tbody>
</table>

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
Interface IP address: 10.1.1.3 10.1.1.4
RP/0/RSP0/CPU0:router-arizona# ping mpls nil-fec labels 16005,16007 output interface GigabitEthernet0/2/0/1 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:

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


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
Examples: LSP Ping and Traceroute for Nil_FEC Target